Mining a large shopping database to predict where, when, and what consumers will buy next

Bantu Halam

Dissertation presented in fulfillment of the requirements for the Master of Science in Mathematical Statistics in the Department of Statistical Sciences at the University of Cape Town.

Supervised by Associate Professor Ian Durbach

July 2019

Abstract

Retailers with electronic point-of-sale systems continuously amass detailed data about the items each consumer buys (i.e. what item, how often, its package size, how many were bought, whether the item was on special, etc.). Where the retailer can also associate purchases with a particular individual for example, when an account or loyalty card is issued, the buying behaviour of the consumer can be tracked over time, providing the retailer with valuable information about a customer's changing preferences. This project is based on mining a large database, containing the purchase histories of some 300 000 customers of a retailer, for insights into the behaviour of those customers. Specifically, the aim is to build three predictive models, each forming a chapter of the dissertation; forecasting the number of daily customers to visit a store, detecting changes in consumers' inter-purchase times, and predicting repeat customers after being given a special offer.

Having too many goods and not enough customers implies loss for a business; having too few goods implies a lost opportunity to turn a profit. The ideal situation is to stock the appropriate number of goods for the number of customers arriving, so you can minimize loss, and maximize profit. To attend to this problem, in the first chapter we forecast the number of customers that will visit a store each day to buy any product (i.e. store daily visits). In the process we also carry out a time-series forecasting methods comparison, with the main aim of comparing machine learning methods to classical statistical methods. The models are fitted into a univariate time-series data and the best model for this particular dataset is selected using three accuracy measures. The results showed that there was not much difference between the methods, but some classical methods slightly performed better than the machine learning algorithms, and this was consistent with outcomes obtained by Makridakis et al. (2018) on similar comparisons.

It is also vital for retailers to know when there has been a change in their consumers purchase behaviour. This change can either be the time between purchases, change in brand selection or change in market share. It is critical for such changes to be detected as early as possible, as speedy detection can help managers act before incurring loses. In the second chapter, we use change-point models to detect changes in consumers' inter-purchase times. Change-point models are approaches that offer a flexible, general-purpose solution to the problem of detecting changes in customer historic behaviour. This multiple change-point model assumes that there is a sequence of underlying parameters, and that this sequence is partitioned into contiguous blocks. These partitions are such that the parameter values are equal within, and different between blocks, whereby a beginning of a block is considered to be a change point. This change-point model is fitted to consumers inter-purchase times (i.e. we model time between purchases) to see whether there were any significant changes on the consumers buying behaviour over a

one year purchase period. The results showed that, depending on the length of the sequences, minority to a handful of customers do experience changes in their purchasing behaviours, with the longer sequences having more changes than the shorter ones. The results seemed to be different to those obtained by Clark and Durbach (2014), but analysing a portion of sequences of same lengths as those analysed in Clark and Durbach (2014), lead to similar results.

Increasing sales growth is also vital for retailers, and there are various possible ways in which this can be achieved. One of the strategies is what is referred to as up-selling (whereby a customer is persuaded to make an additional purchase of the same product or purchase a more expensive version of the product.) and cross-selling (whereby a retailer sells a different product or service to an existing customer). These involve campaigning to customers and sell certain products, and sometimes include incentives in the campaign with the aim of exposing customers to these products hoping they will become repeat customers afterwards. In Chapter 3 we build a model to predict which customers are likely to become repeat customers after being given a special offer. This model is fitted to customers' time between two purchases, which makes the input time-series data, and is sequential in nature. Therefore, we build models that provide a good way for dealing with sequential inputs (i.e. convolutional neural networks and recurrent neural networks), and compare them to models that do not take into account the sequence of the data (i.e. feedforward neural networks and decision trees). The results showed that, inter-purchase times are only useful when they are about the same product, as models did no better than random if inter-purchase times were from a different product in the same department. Secondly, it is useful to take the order of the sequence into account, as models that do this do better than those who do not, with the latter not doing any better than a null model. Lastly, while none of the models performed well, deep learning models perform better than standard classification models and produce some substantial lift.

Acknowledgements

I would like to thank my supervisor, Associate Professor Ian Durbach for introducing me to big data, deep learning methods, and thus data science as this is relevant to my profession and has helped me hone my skills so I can be better equipped for the industry. I would like to thank him for his guidance and support through the learning process.

I would like to acknowledge that the dataset used in this dissertation was obtained from the Kaggle website, and therefore I am grateful to Kaggle for allowing me to have the dataset and use it for my research.

I would like to gratefully acknowledge the scholarships that I received from the African Institute for Mathematical Sciences (AIMS), and UCT Research Fund for my masters degree.

I would like to thank my fiance, family, managers and all loved ones and friends for the support, encouragement, and their belief in me throughout the entire journey. I am genuinely grateful to each and everyone of them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background to the Problem

On a daily basis, retailers get visits from customers, and retailers with electronic point-of-sale systems are able to link customers (i.e. customers that are part of the account or reward base) to purchases. This allows retailers to collect data about the items the consumers buy (including what they buy, how much they spend, how often they buy, etc.), and this leads to having big, detailed data that each store can use for analysis. This huge, detailed amassed data is what is commonly referred to as *big data*, which is a term that describes extremely large data sets (both structured and unstructured) that may be computationally analysed to reveal useful information (i.e. patterns, associations and trends) especially relating to human behaviour for insights that lead to better decisions and strategic business moves. Big data is normally too complex or large for classical data-processing application software to be able to deal with; it is however not the size of the data that is important, but what you do with it as there is a difference between having data, and using it to your advantage.

The practice of transforming existing data into meaningful, actionable insights that businesses can instantly use for decision making is known as data science. Nowadays data science is considered a necessity in most industries, as it is assumed that it is no longer a question whether businesses should make use of the accumulated data, but rather how to make use of it. Data science makes use of complex mathematical and statistical methods and sophisticated systems and environments that are designed to manage big data (e.g. cloud computing). Big data and data science are vital in marketing, and there are many approaches in which these may be applied to enable retailers to be more profitable. These applications include, market budget optimization, marketing to the right audience, matching market strategies with customers, identifying the right channels, lead targeting, and advanced lead scoring.

1.2 Problem Statement

One of the main focuses in marketing is analysing and predicting customer behaviour. Understanding how customers behave helps retailers (i.e. store managers) make better, informed decisions that are data based. By analysing historical purchase data, retailers are able to know what happened, what is happening, what might happen, and thus can put measures in place in case such actually do happen (i.e. risk management measures if it is an unfavourable outcome, or opportunity exploitation measures if it is a positive outcome). In this dissertation we apply data science (i.e. machine learning) algorithms and other statistical methods on big data to generate actionable insights for use in marketing industries focusing on customer behaviour.

An ideal situation for any retailer is being able to avoid losses, and to maintain or improve profits. One of the ways to achieve this is to ensure that production or stock equals demand, i.e. have appropriate number of goods for the number of arriving customers. Another important thing for retailers is to keep track of the behaviour of their customers, so they can know when something has significantly changed and act accordingly. Early detection of such changes is vital as it enables store managers to intervene on time (i.e. manage possible loses or exploit positive changes). Retailers are always looking for ways to make more profit, and there are many approaches that can be applied to achieve this, and one of these is generating more profit from the already existing customers. This involves selling customers products they have never bought before or selling them more of the same product they already buy (normally more expensive version of the product). These involve sending customers special offers (e.g. discounts) and invite them to buy a particular product, with the hope that they will continue buying that product after the offer expires.

1.3 Research Objectives

The goal of this dissertation is to examine how data science can be applied to a large transactional dataset to explore questions about consumer behaviour, specifically; how many customers will visit a store, do customers experience significant changes in their purchasing behaviour or whether special offers are able to turn customers into repeat customers. So, in this dissertation we attempt to answer these questions, and thus to solve problems which are some of the problems faced by retailers. To do this, we develop three predictive models, each forming a chapter of its own, and each attempting to solve one of these problems. Thus, the the main objectives that we hope to achieve in this dissertation are:

- In Chapter 2, we will be predicting the number of daily visits each store can expect in the next 3 months, so they can be able to plan accordingly (i.e. have appropriate number of goods for the number of arriving customers)
- In Chapter 3 we develop a change-point model which is a tool that detects changes in

customer behaviour as soon as possible for use by managers. This will allow store managers to act before incurring costs (if the change is unfavourable) or losing out on opportunities to turn profit (if the change is favourable).

- In Chapter 4 we predict which customers are likely to become repeat customers after being given an offer. This will help store managers better target customers for campaigns by only targeting customers with high likelihood of becoming repeat customers.

1.4 Overview

This dissertation constitutes 5 chapters (inclusive of this one which is an introductory chapter). The following three chapters will be focusing on the 3 predictive models; time-series forecasting models will be discussed in Chapter 2, with change-point models covered in Chapter 3, while Chapter 4 will be focusing on predicting repeat customers. Since each chapter uses quite different methods, and in many ways independent of the rest, we will not have single chapters for literature review and results, but instead these will be included separately in each chapter. That is, all further models details, including introduction, literature review, data discussion, methodology, model implementation, results, discussion and recommendations will be discussed within respective chapters for all models. The last chapter will conclude the dissertation by summarizing everything we covered, stating limitations and scope for future work, and providing recommendations.

Chapter 2

Using Time-Series Forecasting Models to Predict Store Daily Visits

2.1 Introduction

2.1.1 Background to the Problem

On a daily basis, retailers get visits from different customers, who purchase goods. Some stores (i.e. furniture, clothing and hardware) have membership accounts, whereby customers can be members of their customer base, and sometimes have credit cards which allow them to buy goods on credit and pay back (with or without interest) over some amount of time. Other stores (mainly supermarkets) have loyalty or rewards accounts, which enable customers to accumulate points which can be converted into cash that they can spend, or have access to discounts and other benefits that non-members do not have. These memberships (i.e. account, loyalty or rewards) enable retailers to collect and store customer data that allows them to track the customers that they have in their database. By tracking this historical purchase behaviour, retailers are better able to plan for the future.

Some of the examples in which historical customer data can be used for planning include the following; if a restaurant knows how many people have been coming in on Fridays after payday historically, they can expect around the same number of people on the same days and prepare enough food and thus make more profit, and on days where they expect fewer consumers prepare equivalent food and avoid losses. On the other hand, if clothing stores know how many of their customers live in certain cities, their age, income and what type of clothing (premium, budget, fashionable, regular, etc.) they normally buy, the store can produce and distribute the merchandise accordingly, such that the costs are minimal, and profits are maximized.

2.1.2 Problem Statement

Having too many goods and not enough customers implies loss for a business; having too few goods implies a lost opportunity to turn a profit. The ideal situation is to stock the appropriate number of goods for the number of customers arriving, so you can minimize loss, and maximize

profit. In other words, if a store could know how many customers to expect each day, then they would avoid losses by having just enough goods for those customers to consume. This is not easy, otherwise all retailers would be profitable and none would experience losses.

This chapter tries to attend to this problem by forecasting the number of customers that will visit a certain store. This can involve a lot of different scenarios, which include; the frequency of purchases (i.e. forecasting the number of customers arriving in the next day, week, month or year) or product purchases (i.e. number of customers to purchase a certain product). In this chapter, we forecast the number of customers that will visit a store each day to buy any product (i.e. store daily visits).

2.1.3 Objectives

In this chapter we will explore, discuss and compare different forecasting methods for univariate time-series data. We will be discussing the forecasting techniques' strengths and weaknesses, and relationships (where applicable) between methods. The chapter will be discussing simple forecasting models, namely; Simple Average (SA), Moving Average (MA) and Weighted Moving Average (WMA). We will also cover exponential smoothing methods (single, double and triple), the more complex method Autoregressive Integrated Moving Average (ARIMA) (we will also discuss the models that make up the ARIMA model in Autoregressive(AR) and Moving Average (MA)). Lastly, we will look at a method of even higher complexity in Artificial Neural Networks.

The primary purpose of this chapter is to compare the performance of the highly rated artificial neural networks with the more traditional ARIMA forecasting model, and even simpler forecasting heuristics such as the averaging and smoothing methods already mentioned.

2.1.4 Outline

This chapter comprises of 6 sections (inclusive of this one which is an introductory section). The following section discusses the forecasting methods. It starts by defining *time-series* data, and statistical forecasting, then moves on to discuss the data that we will be using, and the methodology we will be following. The section also discusses three forecast accuracy measurement metrics, namely; two error magnitude metrics in Mean Absolute Deviation (MAD) and Mean Forecast Error (MFE), and the Diebold-Mariano significance test. The section then concludes by discussing previous work, both on forecasting future purchases and comparing different time series forecasting methods. Section 3 will then discuss the different forecasting techniques that we will be looking at (as mentioned above). In Section 5 we discuss results, compare the different methods based on both statistical significance and error magnitude, and conclude the section by selecting the method that suits our dataset best. The last section summarizes the whole chapter, and concludes with recommendations.

2.2 Forecasting Time-series

2.2.1 Time-series and Forecasting

2.2.1.1 Time-series

A time series is defined as "sequential collection of data observations indexed over time. In most cases, the observed data is continuous and is recorded at a discrete and finite set of equally-spaced points" (Christopher, 1975), therefore it is a discrete-time data sequence. Time series often arises when tracking corporate business metrics (i.e. customer count or profit) or monitoring industrial processes (Neter et al., 1988). For the purpose of this chapter, as mentioned above, the time series is obtained by tracking the number of customers that shop at a particular store on a daily basis, for different stores.

As already mentioned, time-series is time-ordered data, and this data has some underlying patterns that occur over this time. Understanding these patterns is key to analyzing time-series data as the changes (or lack thereof) in the data occur because of the presence (or absence) of these factors. These factors are widely known as time-series components, and are responsible for bringing about changes in a time-series. These components are (Wang and Chaovalitwongse, 2011a):

- Level - the average value in the series.
- Trend - the increasing or decreasing value in the series. In more detail, trend refers to the stable tendency of growth or decline shown in the data. This trend can be either linear or non-linear, and linear and non-linear functions can be utilized to model the trend accordingly.
- Seasonality - the repeating short-term cycle in the series, a pattern that always repeats at a fixed interval.
- Cycle - the gradual, long-term, up-and-down potentially irregular swings in the series. Cycle patterns are similar to seasonal patterns, except that they vary at various intervals.
- Irregularity - sudden changes which are unlikely to be repeated in the series, thus usually make the pattern difficult to identify.

A time series that has unchanging properties is considered to be stationary. Stationary time series tend to be easier to forecast, because their essential characteristics are unchanging. This thus makes stationarity an important factor of time series, such that some forecasting methods assume stationarity of the time-series, while others make the series stationary first before predicting.

2.2.1.2 Forecasting

Forecasting can be defined as the process of predicting some future event(s) based on past and present data, mostly by analysis of trends. It is estimating, at some specified future date some variable of interest. This involves building a model by feeding historical data into the model as an input, then outputting a result, which is referred to as the prediction. This chapter will be forecasting the number of customers that will shop at a particular store on a daily basis in future.

There are two main different types of forecasting, i.e. iterated (1-step ahead) and direct (n-step ahead) (Marcellino et al., 2005). Iterative forecasting predicts one value into the future, then the next actual value (the one predicted at the previous step) is used as a *true observation* when predicting the next value; that process is then repeated till the desired n-values to be predicted into the future are obtained. On the other hand, direct forecasting uses only true observations available at the time of model building to predict the n-values (multiple steps ahead) desired into the future at once, the next true values are not used to fit the model again. There have been debates and investigations about which method is better, with Marcellino et al. (2005) also discussing this, and they concluded that "In theory, iterated forecasts are more efficient if the one-period ahead model is correctly specified, but direct forecasts are more robust to model misspecification", and the iterated forecasts typically outperform the direct forecasts. In this chapter all of our models use the iterated type of forecasting.

2.2.1.3 Forecast Error

Forecast error is the difference between the actual (observed or real) values and the forecasted (predicted) values. This error can be stated as a value or a percentage, and the difference between 100% and this error (i.e. 100 - error) is known as forecast accuracy. It is thus necessary to measure this error as the idea when forecasting is to minimize this forecast error (or maximize the forecast accuracy). Also, in practice it is unlikely to find one model or technique that fits all the data well, so it is always a good idea to fit different models for a particular dataset, and choose the one with the least error. In this chapter we are comparing different forecasting methods, and we will need to measure the forecast accuracy to decide which method is the best. Assuming that we intend to predict h values into the future, we let y_t represent the actual values, and the forecasts be given by $\hat{y}_t$ where $t = 1, ..., h$, then forecast errors are mathematically given by:

$$e_t = y_t - \hat{y}_t. \tag{2.1}$$

There are various metrics that can be used to measure forecast accuracy, and in this chapter we use two error magnitude metrics, and Diebold-Mariano statistic. The magnitude error measures

are; Mean Forecast Error (MFE), which is just the mean of the errors, and Mean Absolute Deviation (MAD), which is the absolute mean of the errors, and are mathematically expressed as:

$$\begin{aligned} MFE &= \frac{1}{h}\sum_{t=1}^{h} e_t \\ MAD &= \frac{1}{h}\sum_{t=1}^{h} |e_t|. \end{aligned} \tag{2.2}$$

The MAD statistic enables us to see how much on average the model predictions deviate from the actual values. On the other hand, MFE tells us exactly by how much on average a particular method tend to over-estimate (negative MFE) or under-estimate (positive MFE). We chose to use MFE and MAD because the combination of these metrics works very well, and gives us a lot of insight, as one measures the biasness (MFE), and the other the actual error (MAD), but both metrics are not model-based and there is no way of telling whether the differences are statistically significant. To take this into consideration, we also use Diebold-Mariano tests to test whether the forecast errors are statistically significant.

The Diebold-Mariano test is a statistical test that compares forecasting accuracy between two forecasting models to assess whether the two models are equally good, or one is less\more accurate than the other. The test was proposed by Diebold and Mariano (1995), and later modified by Harvey et al. (1997), and this chapter uses the latter version of the test. Provided that we have two forecasting methods and their forecasts are given by $\hat{y}_{1t}$ and $\hat{y}_{2t}$ respectively, then

$$e_{it} = y_{it} - \hat{y}_{it} \quad i = 1, 2. \tag{2.3}$$

The loss associated with forecast from method i is assumed to be the function of e_{it}, the forecast error (Triacca, 2011), and is denoted by $g(e_{it})$. This loss function, $g(e_{it})$ is typically the square (squared-error loss) or the absolute value (absolute error loss) of e_{it}. From the loss function, we define a loss differential as $d_t = g(e_{1t}) - g(e_{2t})$. The two forecasts would then be equally accurate if the expectation of d_t is equal to zero for all t. The null hypothesis we test in this chapter is thus $H_0 : E(d_t) = 0 \ \forall \ t$ versus the alternative hypothesis that the second model is less accurate than the first, $H_1 : E(d_{1t}) < E(d_{2t}) \ \forall t$, where d_{1t} is the loss differential for the first forecast method, and d_{2t} the loss differential for the second forecast method.

It can be shown under H_0 that

$$\frac{\bar{d}}{\sqrt{\frac{2\pi f_d(0)}{\gamma}}} \to N(0, 1), \tag{2.4}$$

where $\bar{d} = \sum_{t=1}^{T} d_t$ is the sample mean of the loss differential (d_t), and $f_d(0) = \frac{1}{2\pi}\left(\sum_{k=-\infty}^{\infty} \gamma_d(k)\right)$ is the spectral density of the loss differential at frequency 0, where $\gamma_d(k)$ is the auto-covariance of the loss differential at lag k. So, for forecasts where $h = 1$ (h being the number of forecasts in future), the Diebold-Mariano(DM) statistic is:

$$DM = \frac{\bar{d}}{\sqrt{\frac{2\pi \hat{f}_d(0)}{\gamma}}}, \tag{2.5}$$

where $\hat{f}_d(0)$ is a consistent estimate of $f_d(0)$, defined by $\hat{f}_d(0) = \frac{1}{2\pi}\hat{\gamma}_d(0)$. Hence for $h \geq 1$ the DM statistic is given by:

$$DM = \frac{\bar{d}}{\sqrt{\frac{\hat{\gamma}_d(0) + 2\sum_{k=1}^{h-1} \hat{\gamma}_d(k)}{T}}}. \tag{2.6}$$

In practice, $2\pi f_d(0)$ is adequately estimated by $\sum_{k=-M}^{M} \hat{\gamma}_d(k)$ in many cases, where $M = T^{1/3}$, thus

$$DM = \frac{\bar{d}}{\frac{\sqrt{\sum_{k=-M}^{M} \hat{\gamma}_d(k)}}{T}}. \tag{2.7}$$

The test statistic, DM is asymptotically $N(0, 1)$ distributed under the null hypothesis. Therefore, if the computed $|DM| > z_{\alpha/2}$ (with $z_{\alpha/2}$ being the upper z-value from the standard normal table that corresponds to half of the desired α level of the test) then the null hypothesis of no difference will be rejected.

The DM test is considered model-free, and its main advantages are that it can be easily "applied to non-quadratic loss functions, multi-step forecasts and forecast errors that are non-Gaussian, non-zero mean, serially correlated and contemporaneously correlated" (Shen et al., 2019). However, the DM test assumes stationarity of covariance (and other regularity conditions) on d_t, hence the main condition is that it should be applied in large samples, as it tends to be oversized in small samples, and thus reject the null hypothesis too often. The DM test modification of small-sample was proposed by Harvey et al. (1997), concerning an unbiased (approximately) estimate of the mean loss differential variance. This is possible when the assumption of zero autocorrelations at order h and beyond holds for the errors of h-steps-ahead forecasts, and the accuracy of the forecast is measured in terms of the mean squared error (Shen et al., 2019). Harvey et al. (1997) suggested obtaining improved small-sample properties by (a) making a bias correction to the DM test statistic, and (b) comparing the corrected statistic with a Student-t distribution with (T-1) degrees of freedom, rather than the standard normal (Luger, 2005). The corrected DM statistics, which is being used in this chapter is thus expressed as:

$$HLN - DM = \sqrt{\frac{T + 1 - 2h + h(h-1)}{T}} DM. \tag{2.8}$$

2.2.2 Data and Methodology

2.2.2.1 Data

This project is based on mining a large database, containing the purchase histories of some 300 000 customers of a retailer, for insights into the behavior of those customers. The data used is obtained from the Kaggle website and was used for the "Acquire Valued Shoppers Challenge" competition. The data comprises of different datasets, but the one which we will be using in this chapter is the transactional dataset, which contains transaction history for all customers for a period of at least 1 year prior to the customer being offered incentives. For this chapter we extracted a subset of the full dataset consisting of 44 stores for whom at least one customer shopped at the store every day between 2 March 2012 and 1 March 2013. Our data consists of daily number of shoppers at each of these stores over this one-year period. This chapter will then focus on forecasting the daily customers for each store. Also, for the purpose of demonstration, we will only use data of a single store to graphically demonstrate the difference between the performance of different methods. We will however include a tabular form of results at the end of the chapter that will include results for all stores.

2.2.2.2 Methodology

The methodology covers the implementation of average methods (simple, moving and weighted), exponential smoothing methods (single, double and triple), ARIMA model and neural networks. We split our dataset into training and test datasets, the training dataset will consist of the first 75% of each store's data, whereas the other 25% will form the test set. We will then use the training dataset to fit models, and forecast ahead for all time points covered by the test data using the iterated type of forecasting. The forecasted values will then be compared to the actual values to measure forecasting accuracy. The forecasting accuracy will be measured using the metrics already discussed above, namely; MFE, MAD and DM statistic. Average methods were implemented in Python. Exponential smoothing methods were implemented in R using the $HoltWinters$ function from the *stats* package. ARIMA model was also implemented in R using the *Arima* function from the $forecast$ package. Neural Networks were also implemented in R, using *keras* package.

2.2.3 Review of Past Forecasting Work

2.2.3.1 Forecasting Future Purchases

Customer Behaviour Analysis or Customer Base Analysis is the analysis whereby historic purchase behaviour of a customer is analyzed in order to understand the customer's current behaviour, or predict the future behaviour. The concept of analyzing customer historic data in order to predict future behaviour has been researched in many fields of academia and imple-

mented in practice as well (Zitzlsperger et al., 2015). These predictions include knowing when the customer is likely to buy next, probability of a customer lapsing (customer being inactive), cross-sell (whereby a retailer sells a different product or serviceto an existing customer) and up-sell (whereby a customer is persuaded to make an additional purchase of the same product or purchase a more expensive version of the product) probabilities, and knowing which customers are likely to be retained.

Several studies have been published on predicting future customer behaviour (Schmittlein and Peterson (1994); Sharad et al. (2008); Ehrenberg (1959); Zitzlsperger et al. (2015)). A similar study to ours (Lee et al., 2016) used different forecasting methods to predict future daily customers for 5 different restaurants, with the idea being to let the restaurants managers know how busy their restaurants are likely to be the following day, so that they can use the forecasted value as a mental preparation. The study focused on 1-time step forecasting, but 2-step and 3-step ahead forecasts also performed well, so all forecasts would be provided to the managers, and they would decide for themselves whether to use the multi-step ahead forecasts.

2.2.3.2 Comparing Different Forecasting Methods

There are many papers that have been published on comparing different forecasting methods for time-series data. To name a few, Smith and Agrawal (2015) compared various forecasting methods on patent group data. Wang and Chaovalitwongse (2011b) compared limitations and advantages of different forecasting methods, and Udom and Phumchusri (2019) compared different forecasting methods for sales forecasting distributor in the plastic industry.

Most recently though, Makridakis et al. (2018) compared classical methods to machine learning methods, extending a study published by Wang and Chaovalitwongse (2011a), where they compared different machine learning forecasting methods. The motivation behind the study was to demonstrate the performance capability of different machine learning methods as compared to traditional statistical forecasting methods. These were compared on a diverse and large collection of forecasting problems of univariate time series. The number of papers claiming that machine learning and deep learning methods offer superior results for time series has been increasing with little objective, this study was thus in response to these claims.

Makridakis et al. (2018) compared 8 classical methods, and 10 machine learning methods, and the 8 classical methods discussed were; Naive 2, Single Exponential Smoothing, Double Exponential Smoothing, Damped Exponential Smoothing , Average of (Single, Double and Damped) exponential smoothing methods, Theta method, ARIMA model and ETS (Error, Trend and Seasonality) method. These were compared to the following 10 machine learning methods; Multi-Layer Perception (MLP), Bayesian Neural Network (BNN), Radial Basis Functions (RBF), Generalized Regression Neural Networks (GRNN), K-Nearest Neighbor Regression (KNN), Cart Regression Trees (CART), Support Vector Regressions (SVR), Guassian Process (GP), Recur-

rent Neural Networks (RNN) and Long-Short Term Memory network (LSTM).

The analysis was performed on univariate data, and both one-step and multi-step ahead forecasting were performed and the methods were compared on both types of forecasting. The outcome from this study was that classical methods such as ETS and ARIMA outperform machine learning methods for one-step ahead forecasting on univariate datasets. Also, on univariate datasets, machine learning methods are outperformed by classical methods such as Theta and ARIMA models for multi-step ahead forecasting. The RNN and LSTM were among the least accurate methods, despite these methods being highly regarded in the recent years, with MLP and BNN outperforming the other machine learning methods. Because of these findings, Makridakis et al. (2018) suggested the use of classical methods before more elaborated and complicated methods are explored, that the results from the simpler methods should be used as a baseline that more complicated methods must better or clear in order to justify their usage. They also acknowledged the potential of the machine learning methods, but suggested that further research is required to make sure that these methods become more accurate, require less computing time, and be less of a black-box.

2.3 Forecasting Methods

2.3.1 Simple Average

For the simple average method, the future values are equal to the mean of historical data. Mathematically, this method can be expressed as:

$$\hat{y}_{t+1} = \frac{1}{n}\sum_{i=1}^{n} y_i, \tag{2.9}$$

where n is the total number of historical values, y_i the observation at time i and $\hat{y}_{t+1}$ the predicted future value for time $t+1$.

Figure 2.1 shows forecasts obtained from a simple average method for an example time series. Adding one new value to the series and using it as a true value to predict the next one does not cause any significant change as the new predicted value will always be a similar value to the mean.

2.3.2 Moving Average

The Moving Average (MA) (also known as Simple Moving Average) method, is an improvement over the above discussed Simple Average method, whereby instead of averaging all historical values, the future values are calculated as an average of the last m data points. The idea behind this method is that the values that matter are the recent ones. The mathematical expression of this method is as follows:

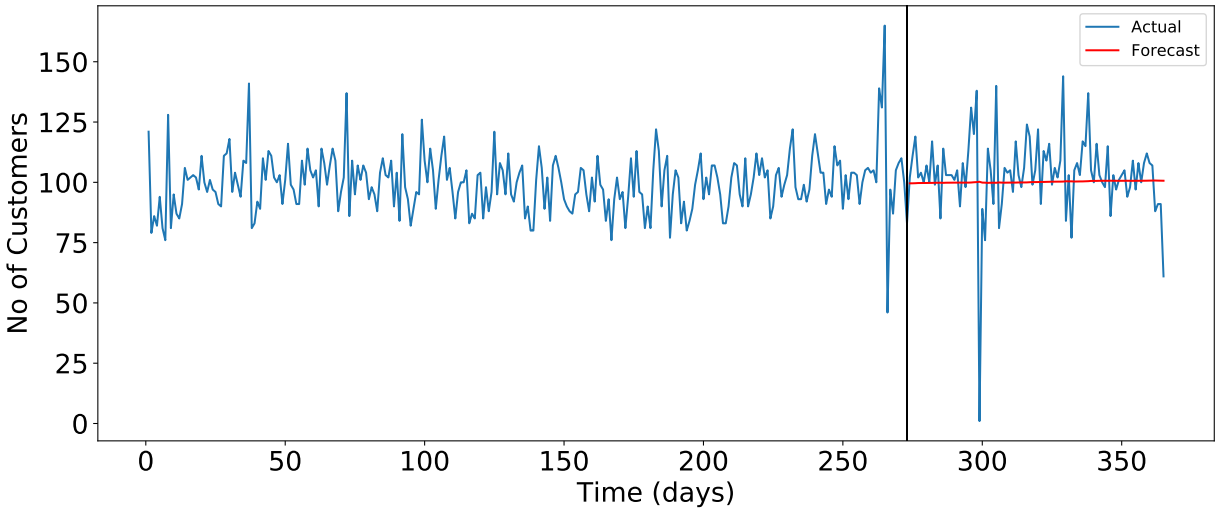


Figure 2.1: Actual data (blue line) and forecasts (red line) obtained from a simple average model. The first 273 days are used to fit the model and n-step ahead forecasts are made for the last 92 for store 71

$$\hat{y}_{t+1} = \frac{1}{m}\sum_{i=1}^{m} y_{t-i}, \tag{2.10}$$

where $\hat{y}_{t+1}$ is the future value, m the number of data points to be averaged, and y the new time series containing the future values.

Figure 2.2 shows results for the moving average method, the next predicted value being the mean of the last m values (in this case m=3).

2.3.3 Weighted Moving Average

The Weighted Moving Average method computes an "average that has multiplying factors to give different weights to data at different positions in the sample window" (Douglass et al., 2012). Instead of assigning equal weights to the last m observations (i.e. for the Simple Moving Average), this method assigns more weighting to the most recent observation (of the last m). This emphasizes the importance of the most recent values. The mathematical expression of this method is as follows:

$$\hat{y}_t = \sum_{i=1}^{m} w_i y_{t-i}, \tag{2.11}$$

where the additional parameter (to those of MA above) w is the series of weights, which should sum up to 1.

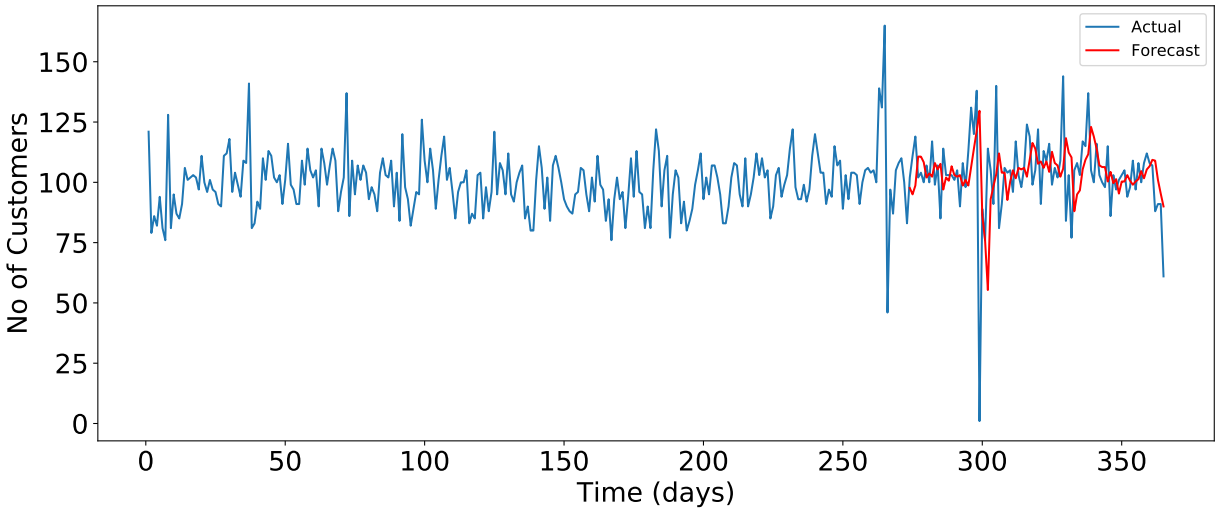


Figure 2.2: Actual data (blue line) and forecasts (red line) obtained from a simple moving average model. The first 273 days are used to fit the model and n-step ahead forecasts are made for the last 92 for store 71

The results for this method are shown in Figure 2.3, with weights assigned to the last 3 observations as follows, 0.5, 0.3 and 0.2 to the t, $t-1$ and $t-2$ values, respectively. The weights were assigned such that the latest observation is the most important, so is the second latest compared to the third latest observation.

2.3.4 Single Exponential Smoothing

The Single Exponential Smoothing method (also known as Brown's Exponential Smoothing) assigns exponentially decreasing weights as the observations get older (Brown, 1963b). This method uses all the observations, but still puts more emphasis on the most recent observations. By using all observations, and also applying weights to all these observations, the single exponential smoothing method aims to predict the short-term future value(s) better than the moving average methods, and in theory should be more accurate. The equation for this method is as follows:

$$\begin{aligned} s_t &= \alpha y_{t-1} + (1-\alpha)s_{t-1} \quad 0 \leq \alpha \geq 1 \\ s_1 &= y_1. \end{aligned} \tag{2.12}$$

where α is the smoothing factor of the level, y_1 the observation at t=1 , and s_t the smoothed value at time t. Smoothing effect for values of α that are small (i.e. close to 0) is bigger compared to that of larger α values, the latter option gives more weight to the most recent observations.

The value of α can be determined (if known) or estimated, and in this chapter α is estimated. This involves minimizing the Mean Squared Error (MSE) between the true and smoothed values

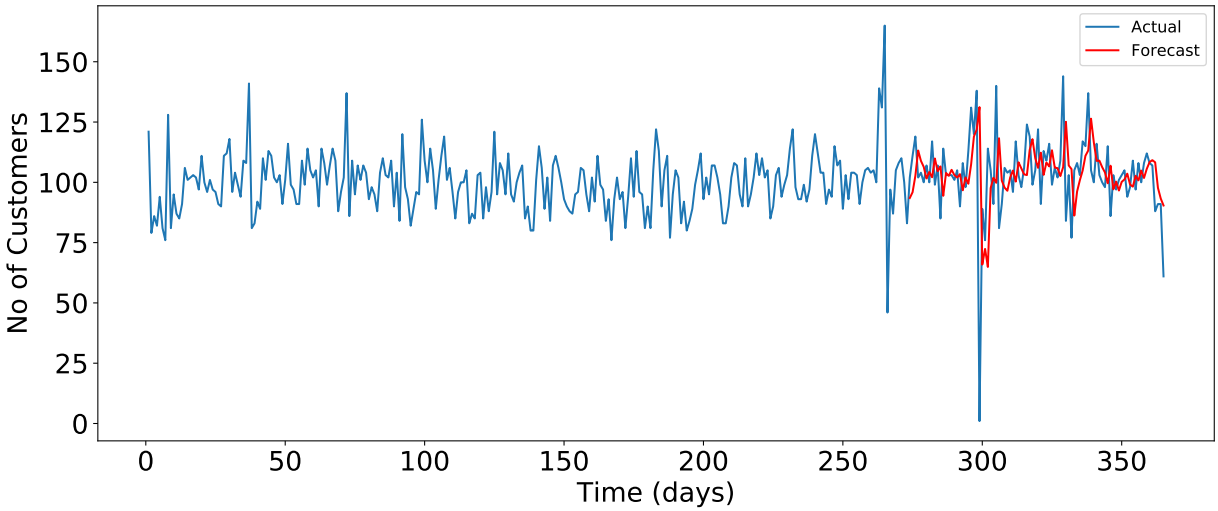


Figure 2.3: Actual data (blue line) and forecasts (red line) obtained from a weighted moving average model. The first 273 days are used to fit the model and n-step ahead forecasts are made for the last 92 for store 71

on the training dataset. A number of different optimization algorithms can be used to estimate α, in this chapter we use quadratic programming. The forecasting equation for Brown's method is then given by:

$$s_{t+1} = \alpha y_t + (1 - \alpha) s_t \quad t > 0. \tag{2.13}$$

Results for this method are shown in Figure 2.4, the forecast line is not as straight as that of the Simple Average, but also does not fluctuate as those of the moving averages. This is because the Brown's method in a way incorporates both methodologies of simple and moving averages, by using all historical data (as per the former) but emphasize importance of the later values by applying weights (as per the latter). Thus in theory, Simple Exponential Smoothing should be better than the simple and moving averages.

2.3.5 Double Exponential Smoothing

The Double Exponential Smoothing method (also known as Holt's Exponential Smoothing), is an extension of the Single Exponential Method, it also estimates the trend of time series on top of the level that is estimated by the Brown's method (Holt, 1957). For this reason, the Holt's method involves using an additional smoothing parameter, β. The smoothing equations for this method are thus expressed as:

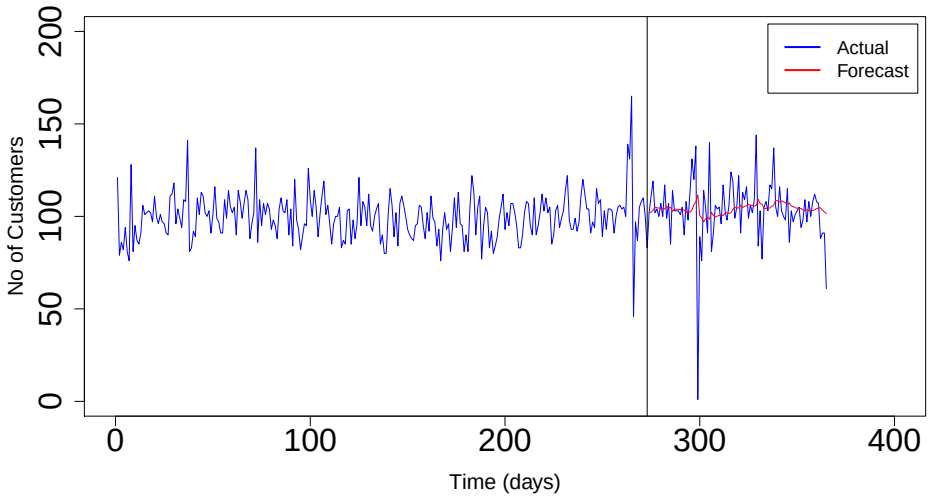


Figure 2.4: Actual data (blue line) and forecasts (red line) obtained from a single exponential smoothing method. The first 273 days are used to fit the model and n-step ahead forecasts are made for the last 92 for store 71

$$\begin{aligned} s_t &= \alpha y_t + (1-\alpha)(s_{t-1} + b_{t-1}) \quad 0 \leq \alpha \leq 1 \\ b_t &= \beta(s_t + s_{t-1}) + (1-\beta)b_{t-1} \quad 0 \leq \beta \leq 1 \\ s_1 &= y_1, \\ b_1 &= y_2 - y_1, \end{aligned} \tag{2.14}$$

where α and β are smoothing factors for level and trend, respectively, with s_t denoting the estimate of the level, and b_t being a trend estimate for the time series. The initial values for the respective smoothing factors are s_1 and b_1. The first smoothing equation directly adjusts s_t for the previous period trend, b_{t-1}. This is done by adding it to the last smoothed value, s_{t-1} (van Greunen, 2015). The second smoothing equation, which is expressed as the difference between the last two values then updates the trend. Similar to Brown's method, the values of the smoothing parameters, α and β are estimated by minimizing the MSE between true values and the observed values. The forecasting equation for this method is then obtained by adding the 2 estimates, s_t and b_t and is expressed as follows:

$$\begin{aligned} y_{t+1} &= s_t + b_t \\ y_{t+k} &= s_t + kb_t, \end{aligned} \tag{2.15}$$

where k is the number of values ahead in time to be estimated.

The results for the DE method in Figure 2.5 show that unlike the Brown's method, Holt's method seems to perform similarly to the moving averages methods.

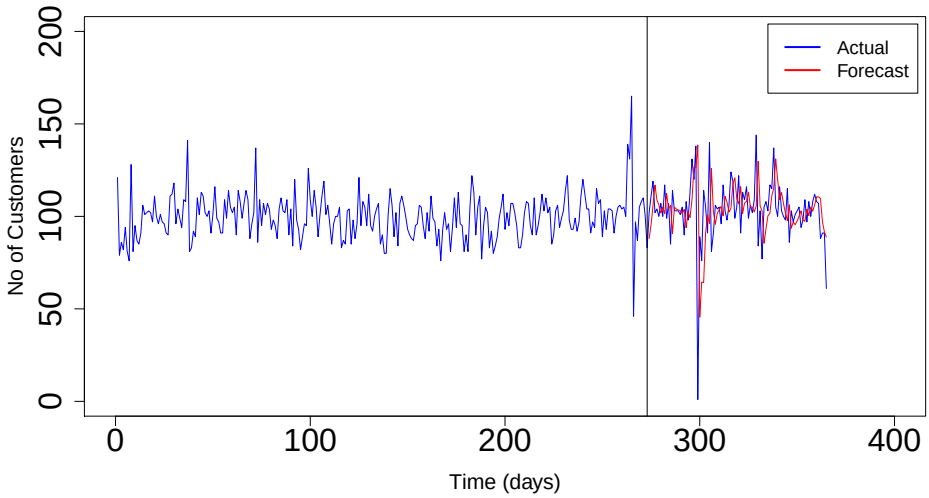


Figure 2.5: Actual data (blue line) and forecasts (red line) obtained from a double exponential smoothing method. The first 273 days are used to fit the model and n-step ahead forecasts are made for the last 92 for store 71

2.3.6 Triple Exponential Smoothing

This method, also known as Holt-Winters Method is the extension of the Double Exponential Smoothing method to also estimate seasonality (Winters, 1960). Therefore, Holt-Winters method applies exponential smoothing to the level, trend and seasonal components. The smoothing formulas for level, trend and seasonal components, respectively are given as:

$$\begin{aligned} s_t &= \alpha \frac{y_t}{I_{t-L}} + (1-\alpha)(s_{t-1} + b_{t-1}) && 0 \leq \alpha \leq 1 \\ b_t &= \beta(s_t - s_{t-1}) + (1-\beta)s_{t-1} && 0 \leq \beta \leq 1 \\ I_t &= \gamma \frac{y_t}{s_t} + (1-\gamma)I_{t-L} && 0 \leq \gamma \leq 1 \\ s_1 &= y_1 \\ b_1 &= \sum_{i=1}^{L} [\frac{y_{L+i} - y_i}{L}] \\ I_1 &= \frac{1}{N} \sum_{j=1}^{N} \frac{y_{L(j-1)+1}}{A_j} \ \forall_i = 1, 2, ..., L, \end{aligned} \tag{2.16}$$

where L is the length of a single season, γ the additional smoothing parameter for seasonality, α and β still the level, and trend smoothing factors, respectively. In s_t, $\frac{y_t}{I_{t-L}}$ deseasonalises the data and leaves only the trend factor and the initial value b_1 to update the process of s_t. Whereas b_t is the smoothed difference between two successive level estimates of the deseasonalised data, I_t is a combination of the most recent observation of the data, divided by the deseasonalised level

estimate, plus the previous seasonal estimate of L samples backward. The last three equations, s_1, b_1, l_i are for setting up initial values of the level, trend and season estimates, respectively. A_j is the average of y for the season corresponding to j index and i is the position within the season. The above season estimate initialization equation produces N seasonal estimates where N is the number of complete seasons in the data (van Greunen, 2015). Similar to Brown and Holt's methods, smoothing parameters are estimated by minimizing MSE. The forecasting formula is then given by:

$$\begin{aligned} y_{t+1} &= \beta(s_t + b_t)I_{t-L+1} \\ y_{t+k} &= \beta(s_t + kb_t)I_{t-L+k}, \end{aligned} \tag{2.17}$$

where k is the number of values ahead in time to be estimated.

Figure 2.6 shows results of the TE method, the forecasts do not fluctuate as those of the Holt's mothod, but also not as steady as those of the Brown's method.

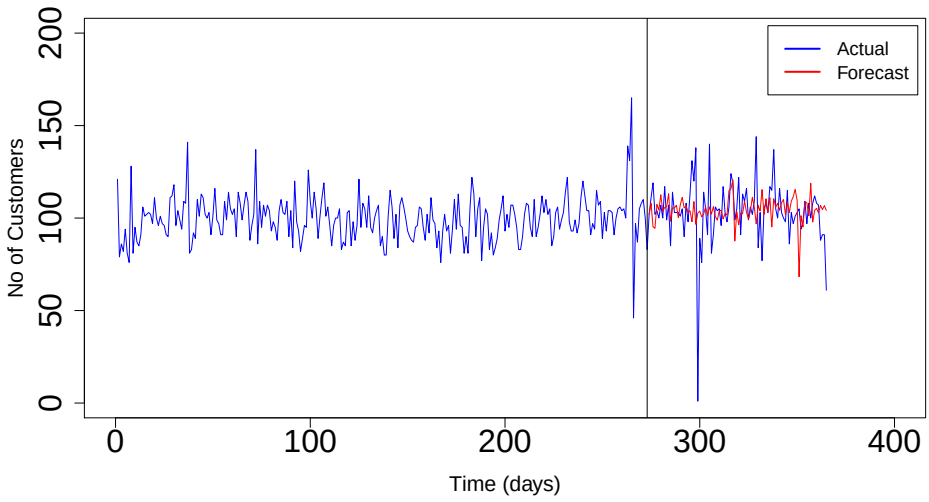


Figure 2.6: Actual data (blue line) and forecasts (red line) obtained from a triple exponential smoothing method. The first 273 days are used to fit the model and n-step ahead forecasts are made for the last 92 for store 71

2.3.7 Auto-Regressive Integrated Moving Average (ARIMA)

ARIMA model is a generalization of an autoregressive moving average (ARMA) model (Valipour et al., 2019), which is also a combination of two models, namely, AR and MA. So it is thus necessary for us to start by discussing the AR, MA and ARMA models before discussing the ARIMA model.

Autoregression works like a linear regression as it also uses a combination of linear parameters

to forecast a future value. The general equation of the AR model of order p (AR(p)) is thus given by:

$$y_t = \theta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t, \tag{2.18}$$

where ϵ is the modelling error, and θ the constant that is expressed as:

$$\theta = (1 - \sum_{i=1}^{p} \phi_i)\bar{y}, \tag{2.19}$$

where $\bar{y}$ is the mean of the time series up to time t.

As we have already seen, variations of moving average (including exponential smoothing) models can be used to mimic the behavior of the most recent past periods, and can provide a good fit for some datasets (de Smith, 2018). These models though are not linked to any statistical models, but can however be specified as statistical/stochastic models that embrace the procedures of moving averages in conjunction with random processes. If we let Z_t be a set of identically distributed and independent random variables with mean of zero and variance with a fixed value that is known, then the previously defined process of $\hat{y}_t$ of the moving average (the weighted moving average in particular) can be written as a moving average of order q, i.e. the equation

$$\hat{y}_t = \frac{1}{n} \sum_{i=1}^{n} w_i y_{t-i} \tag{2.20}$$

becomes

$$\hat{y}_t = \frac{1}{n} \sum_{i=1}^{n} \beta_i Z_{t-i} \tag{2.21}$$

or

$$\hat{y}_t = \beta_0 Z_t + \beta_1 Z_{t-1} + ... + \beta_q Z_{t-q}, \tag{2.22}$$

with mean of zero and variance that is given by:

$$var(\hat{y}_t) = \sigma_t^2 \sum_{i=0}^{q} \beta_i^2. \tag{2.23}$$

Combining these two models by simply adding them together results into a model of order (p, q) with p AR terms and q MA terms. This resulting model known as ARMA is thus given as:

$$y_t = \sigma + \phi_1 t_{t-1} + \phi_2 t_{t-2} + ... + \phi_p t_{t-p} + \epsilon + \beta_0 Z_t + \beta_1 Z_{t-1} + ... + \beta_q Z_{t-q}. \tag{2.24}$$

The problem with ARMA (also AR and MA) is that it assumes stationarity of the time series, which is rarely the case in reality. Trend and seasonality exists in many datasets in practice, and need to be removed before we can apply such a model. The following paragraphs discuss

the extension of the ARMA model that takes into consideration the non-stationarity of the time-series.

As discussed above, ARMA model assumes that the time series is stationary, whereas that is almost always not the case in practice. The ARIMA model attends to this non-stationarity problem by including an initial differencing stage before applying the ARMA model. The "I" in ARIMA refers to the fact that the time series has been initially differenced. Typically, the dataset becomes at least approximately stationary after being differenced once, twice or three times. Differencing up to three times is also advised as over-differencing can pose some dangers, as much of the variations in the data can be lost because of over-differencing (Cochrane, 2018).

ARIMA model, therefore is essentially a combination of ARMA model and differencing of a non-stationary time series, with order (p, d, q). It is therefore necessarily to determine the values of p, d and q before applying the model. We can determine d by finding out how many times we need to difference the time-series before it becomes stationary. The are many methods that are used to determine p and q, and in this chapter we use Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) for q and p, respectively.

The results of the ARIMA model are shown in Figure 2.7.

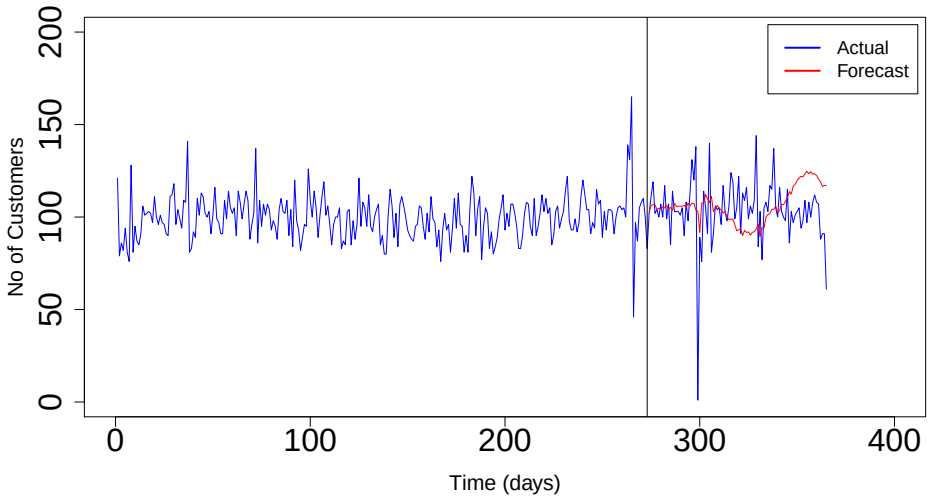


Figure 2.7: Actual data (blue line) and forecasts (red line) obtained from an ARIMA model. The first 273 days are used to fit the model and n-step ahead forecasts are made for the last 92 for store 71

2.3.8 Artificial Neural Network(NN)

Artificial Neural Network (NN) is a machine learning method that uses a large number of computational components to perform computations (van Greunen, 2015). These components,

which are stimulated by an input are known as neurons. NNs are able to compute complex functions and recognize very complicated or detailed pattens within data. "NNs consist of *layers* of neurons with each layer typically fully connected to the next through weighted connections" (van Greunen, 2015). There are 3 different layers involved, input, hidden (with states that are hidden), and output layers. Hidden layers are not a necessity as a neural network can have none, and it is also possible to have multiple hidden layers as well. Input layers feed into hidden layers (if they exist or directly to output layers if no hidden layers exist), then hidden layers feed into output layers, this type of network is known as a Feed Forward Neural Network (FFNN) (Neter et al., 1988). Key Artificial Neural Network (ANN) terms are presented in Table 2.1 (Sambvani, 2018).

It is stated in Haykin (1998) that the artificial neuron model is made up of three distinct parts; *a*) A set of arrows or connections which are characterized by weights, $w_{ij}^{(l)}$, *b*) A linear operation, $\sum$, for combining the inputs as weighted by the respective connections, and *c*) An activation function, f, for limiting the output of the neurons. These components can be mathematically summarized as follows:

$$\begin{aligned} z_j^{(l)} &= \sum_{i=0}^{d^{(l-1)}} w_{ij}^{(l)} a_i^{(l-1)} \\ a_j^{(l)} &= f(z_j^{(l)}), \end{aligned} \tag{2.25}$$

where

$d^{(p)}$ denotes the number of nodes in the p^{th} layer

$a_i^{(r)}$ denotes the value of the output at the i^{th} node of the r^{th} layer

$w_{ij}^{(l)}$ denotes the weight for the connection from the i^{th} neuron in the $l-1^{th}$ layer to the j^{th} neuron in the l^{th} layer.

$z_j^{(l)}$ is the sum of the weighted information, sometimes called the activation, that is input to the j^{th} neuron in the l^{th} layer.

$f(\cdot)$ denotes the activation function.

2.3.8.1 Activation Function

Neural Networks need to make sense of and learn really complicated problems and complex functional mappings that are non-linear between the input and response variables, and Activation Functions are a vital part for this as they introduce to the network non-linear properties. Activation functions define the output of a neuron in terms of the linear combination of the inputs. In other words, they convert an input signal of a node into an output signal, which in turn is used as an input in the next layer. The resulting mapped output values (depending upon

Table 2.1: Glossary of Key Neural Network Terms

Key Term	Explanation	Symbol
Neuron	An information processing unit in a neural network. Each neuron processes some input by applying an Activation Function (defined below) and serves the result of the activation function as its output.	n
Activation Function	The function that we pass the input information through in a neuron that determines the output of the networks. The function is attached to each neuron in the network, introduces to the network non-linear properties so that the network can be able to learn non-linear functional mappings between the input and response variables, and determine whether the neuron should be activated or not based on whether each neuron's input is relevant for the model's prediction.	$f(.)$
Error Function	The function that is being minimized when training the network. This function measures the difference between the desired outcome and the outcome predicted by the network. The size of this difference (as well as the step size) informs how much the parameters at each neuron are changed with each iteration.	E
Layers	Stages of computation of the network (input, hidden, or output)	L
Input Layer	The first layer of a network that contains all input information. Each neuron should represent an input feature. The input layer does not have a bias	L_i
Hidden layer	This is a layer that sits between the input and the output layers. It can have any number of neurons.	L_h
Output Layer	This is the last layer in a neural network. It uses some activation function (e.g. softmax) to produce the models output. Number of outputs desired/required in classification problem determines number of neurons in this layer.	L_o
Gradient Descent	Methodology for calculating how to minimize the cost function by changing weight and bias terms throughout the network.	Δw
Network Learning	The process whereby the model changes weights and bias terms in iterations. The idea behind network learning is to ensure that for each input, the output if not equal to the desired value, is close enough.	
Learning Rate	The speed at which the model changes weights and bias terms with each iteration. By increasing the learning rate, one increases the speed at which a model will learn but also increase the risk that the global minimum will not be found (i.e., the risk that you oscillate on either side of the global minimum because the step size is too large)	η
Batch	Size of the training set that is used in each iteration. A random group of batches are picked during each iteration.	
Weight	Each neuron has weights that multiply each input (i.e. $w1 \times 1 + w2 \times 2 + b$) which goes into the activation function.	w
Bias	Constant added to each input that is used for a neurons activation function	b
Initialization	The initial weights and biases that are used to calculate the outputs of each neuron in the network	

the function) may range between 0 to 1, -1 to 1 etc.. Some of the common activation functions are:

Sigmoid (logistic) Function

Sigmoid activation function (given by Equation 2.26) has an S-shaped curve, and its mapped output ranges between 0 and 1. This is especially useful for models where the output that has to be predicted is a probability (since probabilities range between 0 and 1). The function is differentiable, and monotonic (but the function's derivative is not). Its major problems are; vanishing gradient problem, not zero centered and hence makes optimization harder, Sigmoids saturate and kill gradients, and Sigmoids have slow convergence.

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{2.26}$$

Hyperbolic Tangent Function

The hyperbolic tangent function is simply a scaled version of the sigmoid function, and is linked to it by a transformation; $tan(x) = 2\sigma(2x) - 1$. The output range is -1 to 1, and thus zero centered. Optimization is therefore easier in this method such that in practice it is mostly preferred over Sigmoid function, it however still suffers from vanishing gradient problem. This function is given by:

$$tan(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{2.27}$$

Rectified Linear Unit (ReLu) Function

In the recent years, this function has become popular as it is considered to be simple and efficient and trains faster than $\sigma(z)$ and $tan(x)$. The function's outputs range is 0 to ∞, and the function and its derivative are both monotonic, and it avoids and rectifies vanishing gradient problem. This function is given by:

$$r(x) = max(0, x) = \begin{cases} x & x \geq 0 \\ 0 & elsewhere. \end{cases} \tag{2.28}$$

Softmax Function

A generalization of the sigmoid function, Softmax is commonly used in multi-class classification problems with K distinct classes, and is mathematically expressed as:

$$f_k(x) = \frac{\exp(x_k)}{\sum_{i=1}^{K} \exp(x_i)}, k = 1, 2, 3.., K. \tag{2.29}$$

2.3.8.2 Network Learning

Let us assume that we have an FFNN, whereby a neuron takes an input (x_i), multiplies it with an associated weight (w_i), then maps the weighted sum of the inputs (using the activation function, $\sigma(z)$), and that results to an output, (y). Let us also assume that all the weights (w_i), and biases (b) have been initialized, $\mathbf{T}$ can be defined as training set containing the input values of vector x_j such that each input has a desired output value (d_j). The idea is to ensure that for each input, the output if not equal to the desired value, is close enough. This is done by making small changes to each neuron in order to find weights and biases using the activation function. The process explained above is known as *Neural Networks Learning* (Rumelhart et al., 1986), and can be mathematically approximated to:

$$\Delta y \approx \sum_i \frac{\partial y}{\partial w_i} \Delta w_i + \frac{\partial y}{\partial b} \Delta b, \tag{2.30}$$

where the sum is taken over all i.

2.3.8.3 Error Function

Next we define *error function* denoted as E, which allows us to find weight and biases such that for each training example x_j in $\mathbf{T}$, the desired output d_j is approximated by the network output, y (van Greunen, 2015). One of the most popular error functions when it comes to learning NNs is the *Quadratic Cost Function*, which is equal to the sum of squared errors between actual output and desired output (Rumelhart et al., 1986). The objective is thus to minimize E, and the equation for this function is defined as:

$$E = \frac{1}{2} \sum_j (y_j - d_j)^2. \tag{2.31}$$

To do this, the quadratic cost function uses an approach known as *Gradient Descent*, with an objective of finding a set of weights that minimize E. The gradient descent equation is thus expressed as:

$$\Delta w = -\eta \nabla \mathbf{E}, \tag{2.32}$$

where $\nabla \boldsymbol{E}$ is the matrix of derivatives of E with respect to each weight parameter w_i, and η is known as *learning rate* (a positive value) which controls the learning speed of the NN, with smaller values making the learning slow, and larger values causing the NN to learn quicker.

2.3.8.4 Backward Propagation

Earlier, we discussed how the NN is learned through a process called Neural Network Learning, approximated by Equation 2.30, which in essence is just taking partial derivatives of the output with respect to the weights (and biases). However, in 1986, Rumelhart et al. (1986) proposed a different way of learning networks which actually significantly improved NN learning. They proposed that, instead of learning as explained above (i.e. by taking partial derivatives of the output), partial derivatives of the error function with respect to the weights should be computed instead. That is recursively applying the chain rule for differentiation to iteratively compute gradients of the error function with respect to each of the weights in the network, starting from the output layer and moving backward through the hidden layers of the network (Rumelhart et al., 1986). This then allows the information from the error computed given the outputs in the final layer of the network to flow back through the network by providing a way to calculate the gradients of the error with respect to the weights not only in the output layer but also in the hidden layers which are not directly connected to the output (Rumelhart et al., 1986). This process is known as Backpropagation (Backward-Propagation) and is widely used in the neural network field, mainly because it enables fast network learning, and provides detail into how the weight changes the overall network.

If we let $y = g(x)$ and $z = f(y) = f(g(x))$ then the chain rule of differentiation can be mathematically defined as follows:

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_i}. \tag{2.33}$$

If we define δ_j^l as the error in the j^{th} and l^{th} neuron and layer, respectively, E as the error function, and g as the arbitrary activation function, then the backpropagation computes the error (δ_j^l) for each neuron, which is then used to compute the gradient of the error function (E) with respect to the weights (w_{ij}^l) using the chain rule of differentiation as follows:

$$\frac{\partial E}{\partial w_{ij}^{(l)}} = \frac{\partial E}{\partial a_j^{(l)}} \times \frac{\partial a_j^{(l)}}{\partial w_{ij}^{(l)}} = \delta_j^{(l)} \times x_i^{(l-1)} \; for \; all \; i, j, l. \tag{2.34}$$

The error in the output layer is then defined as:

$$\delta_j^L = \frac{\partial E}{\partial z_j^L} = \frac{\partial E}{\partial a_j^L} \times \frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial E}{\partial a_j^L} \times g'(z_j^L). \tag{2.35}$$

And the error in the layers is defined recursively as:

$$\begin{aligned}\delta_i^{l-1} = \frac{\partial E}{\partial z_j^{l-1}} &= \frac{\partial E}{\partial a_j^{l-1}} \times \frac{\partial a_j^{l-1}}{\partial z_j^{l-1}} \\ &= g'(z_i^{l-1}) \sum_{j=0}^{d^l} \frac{\partial E}{\partial z_j^l} \times \frac{\partial z_j^l}{\partial a_j^l} \\ &= g'(z_i^{l-1}) \sum_{j=0}^{d^l} \delta_j^l \times w_{ij}^l.\end{aligned} \tag{2.36}$$

The summarized backpropagation algorithm is as follows, and can be graphically presented as shown in Figure 4:

start algorithm:

(1) **Input**

Vector of inputs $\mathbf{x}$ with a corresponding ground-truth label y

(2) **Input**

Neural network model function $f(\mathbf{x}; W)$ parameterized by $\mathbf{W}$

(3) **Input**

Activation function $g(\cdot)$

(3) **Input**

Error function E

(5) Compute current prediction

using forward propagation $\mathbf{a}^L = \hat{\mathbf{y}}_i = f(x_i; \mathbf{W})$

(6) Compute error $E(\hat{\mathbf{y}}_i, \mathbf{y}_i)$

(7) Compute error for each neuron in output layer $\boldsymbol{\delta}^L = \nabla_{a^L} E \odot g'(\mathbf{z}^L)$

(8) Compute weight gradients in output layer $\nabla_{W^L} E = \delta^k \mathbf{a}^{k-1}$

(9) **for** l in L - 1, L - 2, . . . , 1 **do**

(10) Compute error for each neuron in l^{th} layer $\boldsymbol{\delta}^l = \mathbf{W}^{l+1} \boldsymbol{\delta}^{l+1} \odot g'(\mathbf{z}^l)$

(11) Compute weight gradients for neurons in l^{th} layer $\nabla_w^l E = \boldsymbol{\delta}^l \mathbf{a}^{l-1}$

(12) **return** Weight gradients $\nabla_W E$

where $\odot$ is used to indicate element-wise multiplication. In Figure 2.9 are the results for the Neural Network method.

2.3.9 Results and Discussion

Table 2.2 contains overall (all stores combined) MFE and MAD statistics for all our 8 methods. Tables 2.3 and 2.4 contain the number of stores for which each method was the best on MFE and MAD, respectively. Tables 2.5 and 2.6 contain MFE and MAD results for all individual stores, respectively. Figures 2.10 and 2.11 show the number of stores that fall within each MFE and MAD band, respectively for each method. The MFE bands were created by grouping stores by

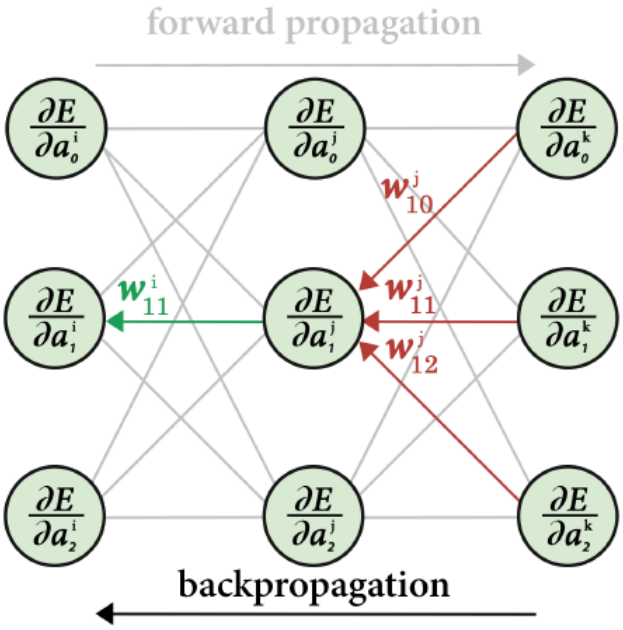


Figure 2.8: Graphical representation of a feedforward neural network architecture

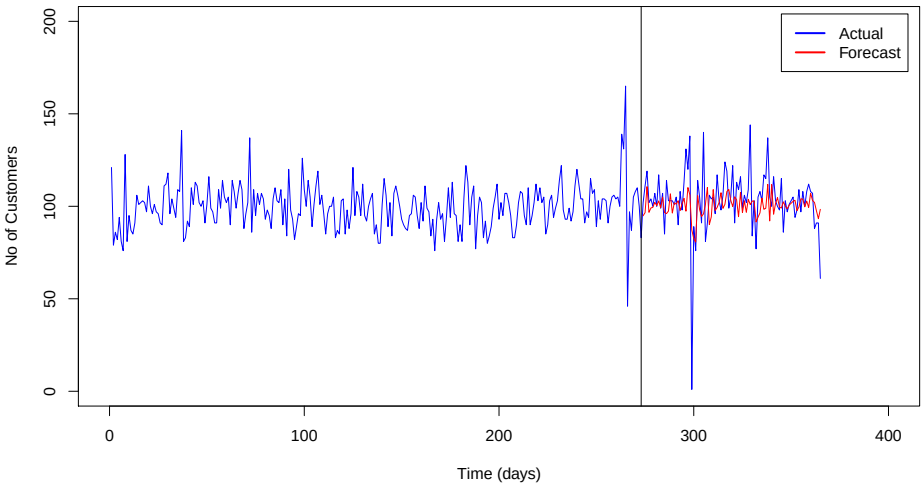


Figure 2.9: Actual data (blue line) and forecasts (red line) obtained from a neural network model. The first 273 days are used to fit the model and n-step ahead forecasts are made for the last 92 for store 71

MFE into 4 different groups; stores with MFE less than -10 were grouped together, then those with MFE between -10 and 0 were grouped together. The last two groups consists of stores with MFE between 1 and 10, and those with MFE greater than 10, respectively. The MAD bands were created in a similar fashion as well, with the four resulting groupings being as follows; the first with stores that have MAD less than 25, then those with MAD between 25 and 49, and the last two consisting of stores with MAD between 50 and 75, and those with MAD greater than 75, respectively.

As already discussed in Section 2.2.1.3, MFE stands for Mean Forecast Error, and it is a measure of whether a method on average overestimates or underestimates the actual values. Negative MFE means the method is overestimating and positive MFE means the method is underestimating. MFE of 0 means the method is unbiased (i.e. underestimate and overestimates by the same margin on average). On the other hand, MAD (Mean Absolute Deviation) is a measure of exactly how much the method's forecasts deviate from true values, a true forecast error. MAD of 0 would imply perfect forecasts. So, an ideal situation would be having both MFE and MAD close to 0, as that would imply that the method is both unbiased, and does not deviate much from the true values.

Table 2.2: Accuracy metrics (MFE and MAD) results for overall stores

Metric	Method							
	SA	MA	WA	SE	DE	TE	AR	NN
MFE	45	1	1	4	1	-7	1	28
MAD	96	99	94	82	87	98	77	82

Table 2.3: Number of stores for which each method was the best on MFE

Metric	Method							
	SA	MA	WA	SE	DE	TE	AR	NN
Clear Wins	0	2	1	0	6	2	3	0
Tied Wins	3	28	27	15	19	8	10	3
Total Wins	3	30	28	15	25	10	13	3
Total Losses	41	14	16	29	19	34	31	41

The main main message from the results is that there is no clear winner between the methods, but that the simpler methods tend to do at least as well, and sometimes a little bit better, than the complex ones. The results have both differences and similarities with those presented by (Makridakis et al., 2018). While our results show a few differences between the methods, in Makridakis et al. (2018), classical methods clearly outperformed the machine learning methods. One could however argue that our most complex method (ANN) is not as complex as the methods evaluated by Makridakis et al. (2018) (i.e. complexity somewhere in between the simple and

Table 2.4: Number of stores for which each method was the best on MAD

Metric	Method							
	SA	MA	WA	SE	DE	TE	AR	NN
Clear Wins	0	0	0	2	0	0	19	6
Tied Wins	8	2	2	16	5	2	15	8
Total Wins	8	2	2	18	5	2	34	14
Total Losses	36	42	42	26	39	42	10	30

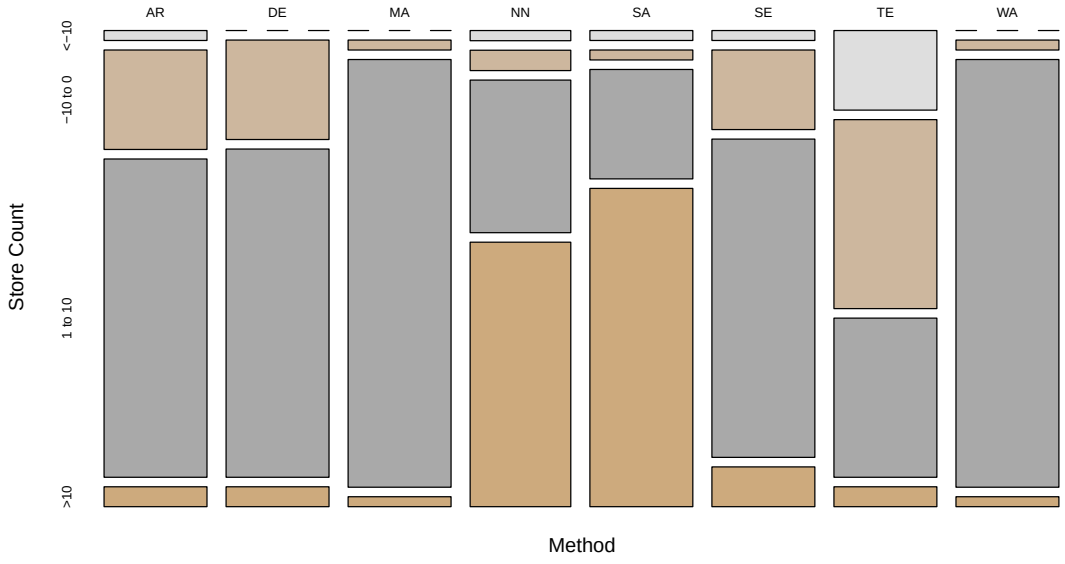


Figure 2.10: Count of number of stores per MFE band for each method

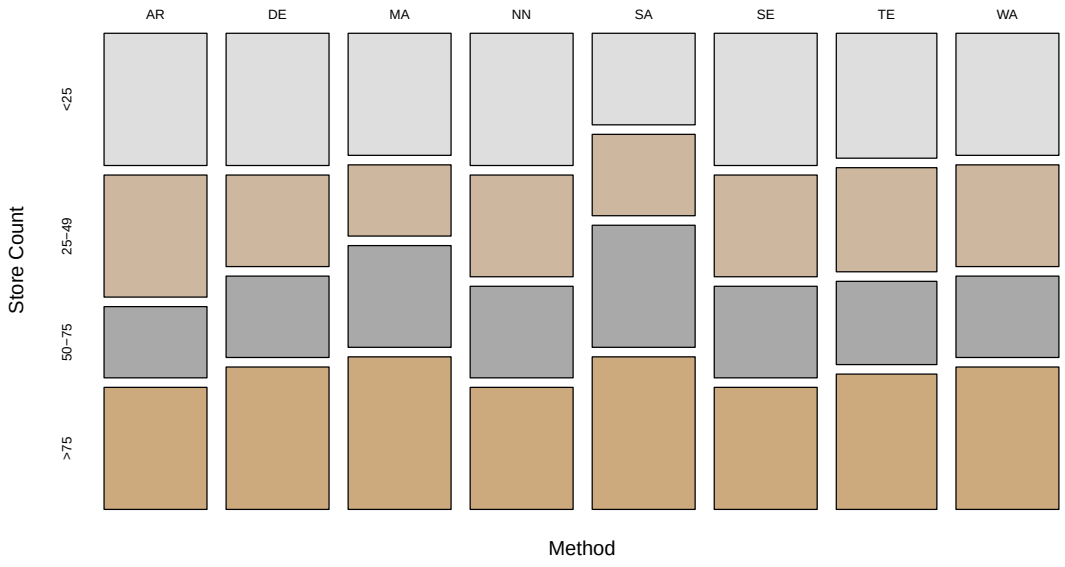


Figure 2.11: Count of number of stores per MAD band for each method

complex methods), hence the somehow different results. However, there were some similarities as well, as some of our better performing methods performed better on their dataset as well.

MFE and MAD results in Table 2.2 show that SA on average under-estimates the most (45), followed by NN (28), while TE is over-estimating the most, by 7 on average. MA, WA , DE and AR tend to under-estimate and over-estimate by almost the same margin, hence the MFE of 1, which makes these methods appear unbiased (i.e. MFE = 0 says that the method is unbiased, that the average forecast error is neither too high or too low). When we look at MAD results we see that there is no MAD close to 0, which would imply perfect fit. MA has the biggest MAD (99), with AR having the least MAD (77). The results we see in Table 2.2 are also visible when we look at at the MFE results in Figure 2.10, with MA and WA having almost all their MFE values for stores falling in the 1 - 10 band (i.e. the dark gray bar), with SA and NN dominating the >10 band (dark brown bar), whereas TE has the most number of stores in the <-10 band (light gray bar). Similarly when we look at MAD results in Figure 2.11, we see that SA and MA has the smallest of the two top bars (fewest stores in the lowest bands), while AR has the smallest of the bottom two bars (fewest customers in the higher bands). Therefore, based on the overall MFE and MAD statistics AR is the best method with MFE of 1 and MAD of 77. Also, results in Table 2.4 show that based on MAD, AR was clearly the best method for 19 (43%) stores, and tied as the best for 15 (34%), and hence being the best for a total of 34 (77%) stores. SE follows second with 18 (41%), and NN in third place with 14 (32%).

As mentioned above, a preferable result would be whereby we have both MFE and MAD closer to 0, so this means the best method will be the one that has the smallest MAD and MFE, and in our case this method is AR for our data. This implies that AR is the most unbiased method, and that its forecasts are the closest to the actual values. AR could be our best method because, it is the only method that takes the stationarity of the data into consideration, and makes sure that the data is stationary before fitting the model.

Table 2.7 contains the Diebold-Mariano test results for all stores. Result column has 3 possible values, Yes if Method 1 is more accurate than Method 2, $Same$ if there is no significant difference, and No if Method 1 is less accurate. The last column is a modified and more lenient form of the Results column, the Result column takes both the Statistic and $P - value$ into consideration when making the decision, whereas Modified only looks at the $P - value$.

The MFE and MAD metrics show us the differences between the methods in terms of forecasting accuracy, but they do not provide a measurer of how significant these differences are as they are not model based. The DM test on the other hand tests the significance of the differences between the methods, and gives an indication of whether these differences between forecasting accuracies of two methods are statistically significant. This means that, a foretasting method would need to have a significantly better accuracy than all other methods in order to be the best.

The Diebold-Mariano test results in Table 2.7 show that MA is the worst method overall, being less accurate than all methods but TE and NN, with MA level accuracy considered to be statistically indifferent to those of TE and NN. On the other hand, AR, which is the best method is slightly better than SE, the second best method. Result column shows that AR is more accurate than 4 methods (SA, MA, WA and TE), and accuracy level is deemed to be similar to that of SE , DE and NN. On the other hand, SE is more accurate than 3 methods (MA, WA and TE), and similar to SA, DE , AR and NN. Modified column shows that AR and SE are better than all other methods, but just similar to each other and NN. However, when comparing these 2 methods to NN, AR results return the smaller $P - value$ (0.368), compared to that of SE (0.7196). Also, when comparing these two methods to each other, we see that AR $P - value$ (0.0684), is much smaller than that of SE (0.9316), which makes AR just slightly better. Also, in Table 2.2 we saw that AR has the smallest MFE and MAD (1 and 77), compared to those of SE (4 and 82), and that is in line with the conclusions made based on Table 2.7, that AR is the better model than SE, and the best method overall. These conclusion are in line with the conclusions made based on the MAD and MFE results, that AR is the best method, followed by SE and NN in second and third places respectively, with MA being the worst method.

These DM results show that, the differences that we saw earlier when looking at MFE and MAD results, which led to the conclusion that AR is the best method for our dataset were statistically

Table 2.5: Accuracy metrics (MFE) results for all stores

Store	MFE							
	SA	MA	WA	SE	DE	TE	AR	NN
2	20	1	1	1	1	-6	-1	34
4	64	3	3	5	1	-12	8	35
6	41	2	2	7	-6	-18	-4	43
8	5	1	1	1	-1	0	1	7
12	128	0	0	-4	2	-20	-3	39
14	27	0	0	0	-2	-6	2	20
15	59	2	2	3	2	-20	0	52
16	4	2	2	-1	-2	12	3	-1
17	22	1	1	2	0	-5	6	24
18	57	3	2	8	6	6	7	45
20	24	0	1	3	0	-2	3	23
21	161	16	14	89	20	-132	-1	250
23	1	0	0	0	0	-1	0	2
24	67	1	1	4	0	-4	3	2
46	112	3	4	-9	11	-44	-8	100
58	23	0	0	0	0	-1	0	8
64	52	-3	-4	17	-5	7	4	57
71	3	0	0	-1	0	-1	-3	4
88	28	0	1	0	-1	-8	2	25
95	39	3	2	9	1	2	8	39
96	98	1	1	-18	3	-55	-19	119
98	72	1	1	2	-5	-10	3	30
100	8	1	0	2	0	-8	0	13
106	-1	0	0	0	0	0	-1	-2
108	2	0	0	0	0	0	2	5
116	0	0	0	0	0	0	0	2
133	39	0	0	-1	0	-4	-1	17
140	15	1	1	-1	1	-1	0	17
151	36	1	1	1	1	-3	2	11
166	67	0	0	1	0	-3	1	13
191	124	1	1	-6	5	-11	-3	72
205	4	1	1	1	0	1	1	4
211	-24	0	0	12	7	22	13	-19
214	249	7	6	29	3	7	23	52
233	25	0	0	1	0	-6	1	10
278	0	0	0	0	0	0	0	0
306	62	0	0	0	-1	0	0	5
313	0	0	0	0	0	0	0	0
377	96	1	1	7	-1	5	7	26
424	2	0	0	-1	0	-2	-1	4
431	45	2	2	2	1	0	2	18
520	53	0	0	0	-1	-2	0	3
523	32	0	0	0	0	0	0	2
526	48	0	0	1	0	0	1	5
AVG	45	1	1	4	1	-7	1	28

Table 2.6: Accuracy metrics (MAD) results for all stores

Store	MAD							
	SA	MA	WA	SE	DE	TE	AR	NN
2	72	95	88	73	85	83	65	70
4	99	110	98	89	96	98	84	79
6	143	194	177	147	165	178	125	115
8	34	44	40	35	38	38	29	29
12	151	111	105	94	105	110	93	106
14	46	53	49	41	42	47	39	45
15	128	154	136	123	130	138	104	121
16	51	67	61	55	54	62	46	46
17	60	78	71	59	65	70	55	51
18	97	108	99	90	90	99	84	94
20	42	47	44	40	40	45	38	46
21	883	1122	1077	890	949	1155	857	844
23	8	9	9	8	9	9	8	9
24	70	32	30	27	30	30	26	32
46	261	289	275	247	259	284	214	249
58	62	69	66	60	63	70	58	61
64	128	156	152	126	137	143	129	143
71	11	12	13	11	14	12	15	11
88	47	50	46	41	42	47	40	41
95	58	58	54	50	55	56	47	61
96	338	373	349	326	327	387	307	316
98	125	137	126	112	115	132	118	103
100	36	45	42	36	37	44	34	35
106	11	14	13	11	13	14	11	11
108	22	26	27	22	24	25	22	22
116	8	9	9	8	8	9	8	8
133	59	51	49	43	48	48	42	46
140	48	57	55	46	55	52	44	54
151	64	58	57	52	56	64	51	58
166	86	58	58	51	57	61	50	58
191	167	141	136	118	132	145	114	142
205	10	12	12	9	10	11	9	10
211	70	72	69	61	63	71	61	68
214	288	179	171	156	154	194	148	141
233	35	33	31	26	31	28	26	33
278	14	17	16	14	15	16	14	14
306	65	24	24	21	21	24	22	24
313	1	1	1	1	1	1	1	1
377	101	39	35	34	36	36	34	42
424	19	23	22	19	22	20	19	20
431	80	84	86	70	84	85	70	74
520	56	22	22	20	20	20	20	21
523	33	12	12	12	12	14	12	14
526	50	18	18	16	17	18	17	20
AVG	96	99	94	82	87	98	77	82

Table 2.7: Diebold-Mariano test results for all stores (per method comparison)

Method 1	Comparison	Method 2	Statistic	P-Value	Result	Modified
AR	>	SA	-2.4286	0.0076	Yes	Yes
AR	>	MA	-2.5852	0.0049	Yes	Yes
AR	>	WA	-2.4621	0.0069	Yes	Yes
AR	>	SE	-1.4881	0.0684	Same	Same
AR	>	DE	-1.8977	0.0289	Same	Yes
AR	>	TE	-4.0194	0.0000	Yes	Yes
AR	>	NN	0.3371	0.3680	Same	Same
SE	>	SA	-1.7638	0.0389	Same	Yes
SE	>	MA	-2.6398	0.0042	Yes	Yes
SE	>	WA	-2.429	0.0076	Yes	Yes
SE	>	DE	-1.8374	0.0331	Same	Yes
SE	>	TE	-4.3653	0.0000	Yes	Yes
SE	>	AR	1.4880	0.9316	Same	Same
SE	>	NN	0.5817	0.7196	Same	Same
NN	>	SA	-1.0877	0.1384	Same	Same
NN	>	MA	-1.9565	0.0252	Same	Yes
NN	>	WA	-1.7983	0.0361	Same	Yes
NN	>	DE	-1.2309	0.1092	Same	Same
NN	>	TE	-2.4948	0.0063	Yes	Yes
SA	>	MA	-2.1822	0.0146	Yes	Yes
SA	>	WA	-1.9418	0.0261	Same	Yes
SA	>	DE	-1.1628	0.1225	Same	Same
SA	>	TE	-3.3332	0.0004	Yes	Yes
DE	>	MA	-3.0055	0.0013	Yes	Yes
DE	>	WA	-2.6833	0.0037	Yes	Yes
DE	>	TE	-0.8805	0.1893	Same	Same
WA	>	MA	-2.1342	0.0164	Yes	Yes
WA	>	TE	0.6488	0.7418	Same	Same
TE	>	MA	-1.0148	0.1551	Same	Same

significant, as AR forecast accuracy was statistically better than all other methods. This thus substantiate the conclusion made based on MFE and MAD, that AR is indeed the best method.

In practice, only the method with the best accuracy would be used, and this can be done in two ways. The first approach would be to use the best method overall stores (provided that all stores have a centralized analytical team and all models are deployed from this centralized center). The second approach would be to select for each store the method that provides the most accurate results for that particular store. If funds, capacity and systems allow, the latter option would likely be used, as it is more ideal to have store or product specific models in practice, than the models that generalize. To evaluate some of the interesting results with regards to these two approaches, we will look at some examples from the individual store results, the MAD results in particular presented in Tables 2.6 and 2.4. Table 2.4 shows that, AR is the best method for most stores, 34 (77%), which would imply that using AR method for all stores could suffice. However, for some of the stores, AR is not not most accurate method, and perhaps for these stores, the other best performing methods can be applied. For store 98 example (see Table 2.6), though AR is the best model overall, NN has an overall average error that is better by 15 customers than that of AR, suggesting that maybe the NN model can be used instead of the AR method for this store. However, it is worth noting that, the average number of customers visiting this store (98) n a daily basis is 1027, so maybe a difference of 15 customers between the AR and NN model accuracies is not significant, implying that using the AR model for this store as well would not be too costly, and will be more efficient. Another interesting result in Table 2.6 is that, store 6 gets visited by 1353 stores a day on average, and the best method (NN) for this store has an error of 115 customers. That is a true misclassification of just 8%. on average, suggesting that, the model could be a very useful tool for this store.

Also, depending on the type of the store (i.e. restaurant, furniture or service subscription), MFE might be a very useful metric. Overestimation might imply more loss for some stores, whereas underestimation can prove to be costlier for other stores. For example, overestimation for a clothing retailer can be very costly, as the stocks for such stores are mostly seasonal. If a lot of stock is still available at the end of the season, the items would have to sold on sale (this can go as low as less than 50% of the original sale), thus resulting to a sizable loss. On the other hand, under-stocking might not be that costly, as more items can always be stocked if a particular product sells quickly, and customers are likely to wait and come back to buy the same product when it is available again. Stores that cell items on contract or provide other services (such as DSTV installation, cellphone contracts, dry cleaning etc.) for example, might lose more if customers are underestimated than when they are overestimated. This is because for such stores, if a customer comes and cannot be helped because the store is either understaffed or out of stock, customers are unlikely to wait, they are likely to go to a competitor to get the service, meaning that they will not come back for that same service again. So for stores that are

critical about either overestimation or underestimation, an MFE of 0 (or close to 0) would be preferable, if not available, a method that slightly overestimates or underestimates, depending on the nature of the store. Table 2.5 shows that, some stores have an MFE of 0 for all methods (stores 278 and 313), meaning that all models are unbiased for these stores. Another interesting results are that of store 6, which show that NN underestimates by 43 customers, whereas AR overestimates by 4 customers. Even though NN was the best method for this store on overall error (see Table 2.6), if this store considers underestimation costlier, they would use the AR method instead.

2.4 Summary and Recommendations

The objective of this chapter was to predict the number of customers going to a store, and to compare forecasting methods for doing that, especially simple vs complex. This involved fitting 8 different forecasting models on univariate time-series data, comparing their results, and deciding on the method that fits our data the best. This chapter started with an introduction, which stated the background to the problem, then stated the problem we would be trying to solve, followed that by stating what it is that we intend to achieve in the chapter, and concluded with an overview of the chapter. We then went on to define time series data, forecasting, and forecasting error (including how it would be measured). We also discussed the data we would be using, methodology, and the softwares on which the analysis would be performed. Next, we discussed previous work both on forecasting future customers, and comparing different forecasting methods for time-series data. Different methods for forecasting (8 in total) were then discussed and compared, with example results (results of a single store) being presented for each method. The methods discussed were Simple Average, Moving Average, Weighted Moving Average, Single Exponential Smoothing, Double Exponential Smoothing, Triple Exponential Smoothing, ARIMA model and Artificial Neural Networks. For each method we discussed the methodology, estimation and forecasting, then ended with a graph showing the example results of comparison between the actual and predicted future values.

Finally, we presented and discussed results of the whole dataset (results for 44 different stores). The methods were compared firstly on 2 accuracy measures, Mean Absolute Deviation (MAD) and Mean Forecast Error (MFE), then also compared the methods using a statistical model, Diebold-Mariano significance test. Overall results showed that, for our dataset, there were very few differences between the methods, as there was no clear winner. This is in contrast with the results published in Makridakis et al. (2018), which stated that simple methods clearly outperformed the complex ones. For our dataset, AR was the best method, outperforming all other methods, followed by SE and NN, with the worst methods being MA and TE. However, the main purpose of this chapter was to compare the performance of the more complex method artificial neural networks (NN) with the more traditional ARIMA forecasting model, and even

simpler forecasting heuristics such as the averaging and exponential smoothing methods already mentioned. Even though NN was not one of the worst methods, it was still clearly outperformed by ARIMA model, and the Single Exponential (Holt's) method, but NN still outperformed the other 5 methods. These results are a bit different to the ones obtained by Makridakis et al. (2018), where neural networks (RNN and LSTM in particular) were one of the least accurate models and outperformed by most classical methods. The results are also similar in some way though, with ARIMA (our best method) being one of the best methods also in Makridakis et al. (2018). We therefore also agree with the conclusions in Makridakis et al. (2018), that when it comes to time-series forecasting, complex is not always better, and that simpler models should always be used as benchmark that complex methods should clear before being implemented.

Chapter 3

Using Change-point Models to Detect Changes in Consumers' Inter-purchase Times

3.1 Introduction

3.1.1 Problem Statement

It is vital for retailers to know when there has been a change in their consumers purchase behaviour. This change can either be the time between purchases, change in brand selection or change in market share. Many retailers collect data and metrics regularly over time with the aim of tracking customer behaviour, both towards store and towards brands. It is critical for such changes to be detected as early as possible, as speedy detection can help managers act before incurring loses (Kantu and Durbach, 2013). Once changes are detected, interventions can then be made, for example; positive changes can be exploited before competitors notice, and negative changes fixed before they cause significant damage.

To detect these changes, change-point models are used. Change-point model is an approach (or family of approaches) that offers a flexible, general-purpose solution to the problem of detecting changes in customer historic behaviour (Kantu and Durbach, 2013). These types of models can be adapted for use with just about any type of data likely to arise from customer purchases. This is because change-point models are based on examining the underlying statistical distributions that give rise to the observed data (Kantu and Durbach, 2013).

There are two main types of change-point models, single change-point models, and multiple change-point models. The former assumes that there is exactly one change point, and tries to estimate where it is (and perhaps how big the change is). On the other hand, the latter does not make this assumption, and instead also estimate the number of change points (which can include 0). There are various approaches that can be used to estimate the parameters of these models, these approaches include; likelihood based approach (Hinkley, 1970b), penalized likelihood approach (Gupta and Chen, 1996), binary segmentation approach (Scott and Knott,

1974), segment neighborhood search (Braun and Muller, 1998), minimum description length (Davis et al., 2006), and Bayesian methods (Barry and Hartigan, 1993). Single change-point models are generally simpler to build and implement compared to multiple change-point models, as such most multiple change-point models are normally extended versions of the single change-point models, with the Bayesian method being the widely used approach (Killick et al., 2010).

3.1.2 Objective

There may be changes in a customer's historical behavior, and these sudden changes can happen at unknown times. In this chapter, we evaluate the extent of such changes in the sequence of purchases by developing a change-point model. This multiple change-point model assumes that there is a sequence of underlying parameters (i.e. sequence of choice probabilities), and that this sequence is partitioned into contiguous blocks. These partitions are such that the choice probabilities are equal within, and different between blocks, whereby a beginning of a block is considered to be a change point (Barry and Hartigan, 1993).

Change point models that are focusing on changes in a customer's buying behavior consists of two types, i.e. inter-purchase times model (whereby time between purchases is modeled), and brand-choice model (whereby product selection at each purchase occasion is modeled). In this chapter we look into the first type of model, inter-purchase times. The first part will focus on modeling time between store purchases (i.e. customer's time between purchases of any product at any store), and the second part on modeling time between brand purchases (i.e. customer's time between purchases of a particular product at any store).

The rest of the chapter is structured as follows; in the following section we will be discussing previous work on change-point models. In Section 2 we discuss the methodology that will be followed when building the change-point model. This methodology section includes explanations of the data, model description and model implementation. We will then follow with a demonstration section, where we will be illustrating various aspects of the change-point model using simulated data. In Section 4 we will then illustrate the use of the change-point model in practice. The results section will then follow, followed by discussion. The last section is a summary of the chapter, including recommendations.

3.1.3 Previous Work

Change-point model development is not a recent concept, with the early works dating back to the 1950s, with Page (1954a), Shiryaev (1963) and Hinkley (1970a). Change point models can be used for both methodological and practical purposes (Eckley et al., 2011), some of the examples include; validation of model assumptions (Fryzlewicz and Rao, 2011), assessment and monitoring of safety critical processes (Elsner et al., 2004), and the validation of an untested scientific hypothesis (Henderson and Matthews, 1993). Change point models can be applied and used in

a wide range of fields, and has been of interest to various disciplines, including bioinformatic applications (Lio and Vannucci, 2000), finance applications (Spokoiny, 2009), network traffic analysis (Kwon et al., 2006), climatology applications (Jaxk et al., 2007), oceanography applications (Killick et al., 2010) and detection of malware within software (Yan et al., 2019). Eckley et al. (2011) described and compared a number of different approaches for estimating change points, general overview of change point methods was also given by Carlstein et al. (1994), and Chen and Gupta (2000).

In marketing, change point models are used to detect changes in customers purchase behaviour. Clark and Durbach (2014) investigated this concept by developing a change point model to detect change in brand customer-loyalty, in other words, changes in individuals' propensities to purchase a particular product over time. They state that these changes can assume 2 types of forms. The first form being that the probability that a brand is chosen at a particular point in time might be influenced by previous brand choices. The second type of change is for an individual's choice probabilities to undergo random change at unknown times but to be relatively constant between these change-points. Their study was then focusing on this second form, assessing how common changes of this type are in general fast moving consumer goods markets. This involved conducting a change-point analysis on respondent-level binary sequences of purchase data, that is a sequence of 1's and 0's, whereby 1 means a brand of interest was purchased, with 0 meaning another brand was purchased. Their findings were that, the majority of consumers in their panel show little change in purchase probabilities over the medium-term, but that substantive and persistent changes are indicated for a significant minority of consumers.

3.2 Methodology

The goal of this section is to provide an intuitive, accessible explanation of the Bayesian change-point (BCP) models used in this chapter. Most papers on change-point models (Barry and Hartigan (1992); Chib (1998); Loschi and Cruz (2019)) are highly technical and mathematical, and so it can be difficult to gain an understanding for how the models work. The objective of this section is to describe the workings of the change-point model in a way that makes it clear what the model is doing at each step. In doing this we start with a simple example and explain the theory as we need it.

Suppose we have a time series consisting of random variables $X_1, \ldots, X_N$. We assume that these can be split into B blocks of contiguous observations. For example, the first block might contain the first 10 observations $\{X_1, X_2, \ldots, X_{10}\}$; the second block the next 20 observations $\{X_{11}, \ldots, X_{30}\}$; and a third block the remaining observations $\{X_{31}, \ldots, X_N\}$. The key feature defining a "block" is that observations within the same block are independently generated by the same distribution. Observations in different blocks are generated, also independently, from different distributions. For example, the observations in the first block might be generated

from an exponential distribution with parameter $\lambda_{(1)} = 3$; observations in the second block might be generated by an exponential distribution with $\lambda_{(2)} = 5$. The set of all blocks, in this case $\{\{X_1, \ldots, X_{10}\}, \{X_{11}, \ldots, X_{30}\}, \{X_{31}, \ldots, X_N\}\}$ is called a *partition*, denoted by ρ, and the model called a *product partition model*. The underlying assumption is that observations are generated by a process that undergoes instantaneous changes in its parameters.

The task of the change-point model is to estimate, (a) the number of blocks there are; (b) the positions in the time series where the changes occur (i.e. the boundaries between blocks); (c) parameter values within each block.

Rather than presenting the change-point model in general mathematical terms, here we present the model using an example. Suppose that we have observed the following five values:

$$20 \quad 30 \quad 2 \quad 1 \quad 2$$

We assume that all values come from an exponential distribution. To make the extensions introduced by the change-point model absolutely clear, we start off in the typical "no change" setting where we assume that all five observations are drawn from the same distribution $f(x) = \lambda e^{-\lambda x}$. Then the likelihood function for the data is given by $L(\lambda; \mathbf{x}) = \prod_{i=1}^{N} \lambda e^{-\lambda x_i} = \lambda^5 e^{-55\lambda}$, and the log likelihood function by $\ell(\lambda; \mathbf{x}) = 5\lambda - 55\lambda$. This can be maximized to find the maximum likelihood estimate for λ, $\hat{\lambda} = 5/11$, or more generally $\hat{\lambda} = N/\sum_{i=1}^{N} x_i$.

We now introduce the change-point model step by step, adding complexity as we go.

Known number and position of change-points

Suppose that we somehow know that there has been a change between the third and fourth observations, so that the time series can be partitioned into two blocks, $b_1 = \{X_1, X_2, X_3\}$ and $b_2 = \{X_4, X_5\}$. In other words, the partition ρ is assumed known. In doing so we assume that observations in the first block are generated by an exponential distribution with parameter $\lambda_{(1)}$, and observations in the second block are generated by an exponential distribution with parameter $\lambda_{(2)}$.

Because we assume that the two blocks are independent of each other, maximum likelihood estimates can be calculated separately in each block. The likelihood function in block 1 is given by $L(\lambda_{(1)}|\rho) = \prod_{i=1}^{N_1} \lambda_{(1)} e^{-\lambda_{(1)} x} = \lambda_{(1)}^3 e^{-52\lambda_{(1)}}$, giving an MLE $\hat{\lambda}_{(1)}|\rho = N_1/\sum_{i=1}^{N_1} x_i = 3/52$. Here we have explicitly written down that the likelihood function and MLE are conditional on ρ. Similarly, the likelihood function in block 2 is given by $L(\lambda_{(2)}|\rho) = \prod_{i=N_1+1}^{N} \lambda_{(2)} e^{-\lambda_{(2)} x} = \lambda_{(2)}^2 e^{-3\lambda_{(2)}}$, giving an MLE $\hat{\lambda}_{(2)}\rho = N_2/\sum_{i=N_1+1}^{N} x_i = 2/3$.

Note that because blocks are independent the likelihood function for the data as a whole is the product of the likelihood function in each block

$$
\begin{aligned}
L(\boldsymbol{\lambda}|\rho) &= L(\lambda_{(1)}|\rho)L(\lambda_{(2)}|\rho) \\
&= \prod_{i=1}^{N_1} \lambda_{(1)} e^{-\lambda_{(1)} x_i} \prod_{i=N_1+1}^{N} \lambda_{(2)} e^{-\lambda_{(2)} x_i} \\
&= \lambda_{(1)}^3 e^{-52\lambda_{(1)}} \lambda_{(2)}^2 e^{-3\lambda_{(2)}},
\end{aligned}
$$

The calculations above can be used for any number of change-points, so long as their positions are known.

Known number of change-points, unknown position

Now suppose that we know only that there has been a single change, but not where it is. In addition to $\lambda_{(1)}$ and $\lambda_{(2)}$, we also need to estimate the position of the change-point (or, equivalently, estimate the partition). We can do this by noting that there are only four possible positions where the change could have occurred, after any of the first four observations (if the change occurred after the fifth observation then all observations are in the same block, and from the perspective of the model there is no change-point). We can work through the calculations above for each possible partition (i.e. for each possible position of the change-point, as is done in Table 3.1). For each partition, we obtain a maximized value of the likelihood function, that is, a value of the likelihood function under the optimal values of $\lambda_{(1)}$ and $\lambda_{(2)}$ for that partition. We then select the partition that maximizes this likelihood.

Technically, this means taking a profile likelihood approach to estimating ρ. We choose the value of ρ that maximizes $L(\rho) = sup_\lambda L(\lambda, \rho)$. We assume that all partitions are equally likely *a priori* so that same values of ρ maximize $L(\lambda, \rho)$ and $L(\lambda|\rho)$, and that the sum over λ is achieved at the conditional maximum likelihood estimates $\hat{\lambda}_{(1)}$ and $\hat{\lambda}_{(2)}$.

ρ	b_1	b_2	$\hat{\lambda}_{(1)}\|\rho$	$\hat{\lambda}_{(2)}\|\rho$	$L(\hat{\lambda}_{(1)}\|\rho)$	$L(\hat{\lambda}_{(2)}\|\rho)$	$L(\hat{\boldsymbol{\lambda}}\|\rho)$
1	$\{x_1\}$	$\{x_2, \ldots, x_5\}$	0.05	0.11	0.018	3×10^{-6}	6×10^{-8}
2	$\{x_1, x_2\}$	$\{x_3, x_4, x_5\}$	0.04	0.60	2×10^{-4}	0.011	2×10^{-6}
3	$\{x_1, x_2, x_3\}$	$\{x_4, x_5\}$	0.06	0.67	9×10^{-6}	0.060	6×10^{-7}
4	$\{x_1, \ldots, x_4\}$	$\{x_5\}$	0.08	0.50	6×10^{-7}	0.183	1×10^{-7}

Table 3.1: Maximum likelihood estimates for λ, and maximized values of the likelihood function, for different partitions ρ. Values in this table assume that we know that there is a single change-point present.

Unsurprisingly, the change-point analysis puts the most likely position of the change-point after $t = 2$, splitting the data into two blocks $\{x_1, x_2\}$ and $\{x_3, x_4, x_5\}$. A question now is how to use information about the relative likelihoods of different partitions to inform unconditional estimates of $\lambda_{(1)}$ and $\lambda_{(2)}$? There are two possible approaches. One is to use the MLEs associated with the most likely partition, given that X_1 and X_2 are distributed exponentially with parame-

ter $\hat{\lambda}_{(1)} = 0.04$, and the remaining observations are distributed with parameter $\hat{\lambda}_{(2)} = 0.60$. This approach is straightforward but ignores useful information about values of λ in other partitions, as well as the relative likelihoods of those partitions. This is unlikely to matter much here, as the selected partition is much more likely than any other, but this would not always be the case.

The second, preferred approach is to average estimates over partitions, using the relative likelihood of a partition to weight the estimates of λ corresponding to time-points in that partition. Doing so requires a small change in emphasis and notation. Previously we referred to a block such as $b_1 = \{X_1, X_2, X_3\}$ and a parameter estimate within that block, $\lambda_{(1)}$. This was shorthand for saying "observations x_1, x_2, and x_3 are generated from exponential distributions with parameters λ_1, λ_2, and λ_3 respectively, but because these belong to the same block we set $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_{(1)}$". Thus for any block one can write the parameters in a compact form, $\lambda_{(1)}$, or in an expanded form, $(\lambda_1, \lambda_2, \lambda_3)$, and similarly for any partition one can write compactly $(\lambda_{(1)}, \lambda_{(2)})$, or not $(\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5)$, with the blocks in the partition defining which of the λ_i are equal to one another. Table 3.2 expands the block-specific MLEs to the time points covered by each block, and shows how these can be averaged over partitions, weighting by the partition probabilities, to obtain final estimates of λ at each time point. Note that here, because we *know* that one of the four partitions in Table 3.1 must be the correct one, and all four partitions are equally likely, the partition probabilities are simply the likelihoods in the last column of Table 3.1 rescaled to sum to one.

ρ	$\hat{\lambda}_{(1)}\|\rho$	$\hat{\lambda}_{(2)}\|\rho$	$\hat{\lambda}_1\|\rho$	$\hat{\lambda}_2\|\rho$	$\hat{\lambda}_3\|\rho$	$\hat{\lambda}_4\|\rho$	$\hat{\lambda}_5\|\rho$	$\pi(\rho\|\hat{\boldsymbol{\lambda}})$
1	0.05	0.11	0.05	0.11	0.11	0.11	0.11	0.02
2	0.04	0.60	0.04	0.04	0.60	0.60	0.60	0.76
3	0.06	0.67	0.06	0.06	0.06	0.67	0.67	0.19
4	0.08	0.50	0.08	0.08	0.08	0.08	0.50	0.04
Method 2			0.04	0.05	0.47	0.58	0.60	
Method 1			0.04	0.04	0.60	0.60	0.60	

Table 3.2: Expanding the maximum likelihood estimates for λ within each block, to all observations within that block. The final estimates of λ are weighted averages of the conditional MLEs for λ, where the conditioning is on the partition ρ, weighted by the likelihood of the partition. This ensures that the most likely partitions contribute the most to the final estimate of λ. We still assume that we know that there is a single change-point present.

Unknown number of change-points, unknown position

The approach above, which is to use relative likelihoods to either choose between partitions, or to weight partitions when estimating $\boldsymbol{\lambda}$ breaks down when we do not know how many change-points there are. This is because likelihoods cannot be directly compared across partitions with different numbers of change-points; likelihoods tend to be greater for partitions with more change-points, and greatest for a partition that puts every observation in its own block (see Table 3.3).

This problem can be avoided by recognising that not all partitions are equally likely *a priori*, and that for most problems it will be more common to have only a few change-points than many. Intuitively, this is because a change-point should indicate a meaningful, reasonably persistent change that has some kind of operational meaning, rather than a small shift that might under slightly different circumstances be assigned to random chance. Taking this view means taking a Bayesian perspective, and assigning a prior probability to each possible partition. A simple way of doing this is to specify a prior probability p that any time-point will be a change-point. Then the prior probability associated with a partition is determined only by the number of change-points it has. We ignore the possibility of a change-point after the final observation at $t = N$, but change-points can occur at any of the other $N - 1$ observations, so that there are 2^{N-1} possible partitions. If a partition ρ has B blocks, then it has $B - 1$ change-points, and $N - B$ observations that are not change-points (ignoring the final observation). The prior probability of the partition is therefore $\pi(\rho) = p^{B-1}(1-p)^{N-B}$, and the posterior probability of the partition, which replaces the partition probabilities in Table 3.2, is given by $\pi(\rho|\hat{\boldsymbol{\lambda}}) \propto L(\hat{\boldsymbol{\lambda}}|\rho)\pi(\rho)$. Table 3.3 gives results for all possible partitions.

ρ	$\hat{\lambda}_1\|\rho$	$\hat{\lambda}_2\|\rho$	$\hat{\lambda}_3\|\rho$	$\hat{\lambda}_4\|\rho$	$\hat{\lambda}_5\|\rho$	$L(\hat{\boldsymbol{\lambda}}\|\rho)$	$\pi(\rho)$	$\pi(\rho\|\hat{\boldsymbol{\lambda}})$
$\{x_1, x_2, x_3, x_4, x_5\}$	0.09	0.09	0.09	0.09	0.09	4.1×10^{-8}	0.66	0.08
$\{x_1\}, \{x_2, x_3, x_4, x_5\}$	0.05	0.11	0.11	0.11	0.11	5.7×10^{-8}	0.07	0.01
$\{x_1, x_2\}, \{x_3, x_4, x_5\}$	0.04	0.04	0.60	0.60	0.60	2.3×10^{-6}	0.07	0.52
$\{x_1\}, \{x_2\}, \{x_3, x_4, x_5\}$	0.05	0.03	0.60	0.60	0.60	2.4×10^{-6}	0.01	0.06
$\{x_1, x_2, x_3\}, \{x_4, x_5\}$	0.06	0.06	0.06	0.67	0.67	5.7×10^{-7}	0.07	0.13
$\{x_1\}, \{x_2, x_3\}, \{x_4, x_5\}$	0.05	0.06	0.06	0.67	0.67	5.8×10^{-7}	0.01	0.01
$\{x_1, x_2\}, \{x_3\}, \{x_4, x_5\}$	0.04	0.04	0.50	0.67	0.67	2.3×10^{-6}	0.01	0.06
$\{x_1\}, \{x_2\}, \{x_3\}, \{x_4, x_5\}$	0.05	0.03	0.50	0.67	0.67	2.4×10^{-6}	0.00	0.01
$\{x_1, x_2, x_3, x_4\}, \{x_5\}$	0.08	0.08	0.08	0.08	0.50	1.0×10^{-7}	0.07	0.02
$\{x_1\}, \{x_2, x_3, x_4\}, \{x_5\}$	0.05	0.09	0.09	0.09	0.50	1.2×10^{-7}	0.01	0.00
$\{x_1, x_2\}, \{x_3, x_4\}, \{x_5\}$	0.04	0.04	0.67	0.67	0.50	2.3×10^{-6}	0.01	0.06
$\{x_1\}, \{x_2\}, \{x_3, x_4\}, x_5\}$	0.05	0.03	0.67	0.67	0.50	2.4×10^{-6}	0.00	0.01
$\{x_1, x_2, x_3\}, \{x_4\}, \{x_5\}$	0.06	0.06	0.06	1.00	0.50	6.4×10^{-7}	0.01	0.02
$\{x_1\}, \{x_2, x_3\}, \{x_4\}, \{x_5\}$	0.05	0.06	0.06	1.00	0.50	6.5×10^{-7}	0.00	0.00
$\{x_1, x_2\}, \{x_3\}, \{x_4\}, \{x_5\}$	0.04	0.04	0.50	1.00	0.50	2.6×10^{-6}	0.00	0.01
$\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}$	0.05	0.03	0.50	1.00	0.50	2.8×10^{-6}	0.00	0.00
	0.05	0.05	0.45	0.57	0.55			

Table 3.3: Change-point results for all possible partitions. We now weight the conditional MLEs by the posterior probability of the partitions, not by their likelihoods alone; these cannot be directly compared across different numbers of change-points. The remainder of the analysis stays the same.

A further output that can be obtained from Table 3.3 is the posterior probability that a change-point occurs at any time-point $t = 1, \ldots, N$, which may be of managerial interest. The probability that a change-point occurs at time t is easily obtained by summing posterior partition probabilities for the subset of partitions that contain a change-point at time t. If we denote

the set of all partitions for which time t is a change-point as R_t, then the sum we compute is $\sum_{\rho \in R_t} \pi(\rho|\hat{\boldsymbol{\lambda}})$. Thus for example the probability that a change-point occurs at $t = 1$ (between the first and second observations) is obtained by adding the posterior probabilities for $\rho = \{\{x_1\}, \{x_2, x_3, x_4, x_5\}\}$, $\rho = \{\{x_1\}, \{x_2\}, \{x_3, x_4, x_5\}\}$, $\ldots$, $\rho = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}\}$. The respective posterior probabilities that a change-point occurred at $t = 1, 2, 3, 4$ are 0.106, 0.716, 0.233, and 0.119 respectively.

An almost fully Bayesian change-point model

The change-point model as described above is already partially Bayesian, in that it includes prior probabilities on the set of partitions. A fully Bayesian change-point model has two further advantages, the ability to model prior knowledge about λ and p, and avoiding having to compute the likelihood for all possible partitions. The first of these is convenient but the second is essential for longer time series, as the number of partitions is 2^{N-1}, so increases exponentially with the number of observations and can quickly become extremely large.

For the time being we do not put a prior distribution on p (the prior probability that any time-point is a change-point) but assume that we know this value. This simplifies the illustration below and requires only a small extension to relax. Prior knowledge about λ is most simply modelled with a gamma distribution, which is a conjugate prior for an exponential likelihood, i.e. $\pi(\lambda) = \phi^\alpha \lambda^{\alpha-1} \exp(-\phi\lambda)/\Gamma(\alpha)$, for shape and scale parameters α and ϕ. Then a prior-to-posterior analysis gives:

$$\begin{aligned}
\pi(\lambda|\mathbf{x}) &\propto L(\lambda)\pi(\lambda) \\
&= k\left[\prod_{i=1}^{N} \lambda e^{-\lambda x}\right]\left[\frac{\phi^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\phi\lambda}\right] \\
&= k^*\lambda^{N+\alpha-1}e^{-\lambda(\sum_i x_i+\phi)} \\
&\sim \text{gamma}(N+\alpha, \sum_i x_i + \phi),
\end{aligned}$$

where we absorb terms not involving λ into k and k^*. Note that, once we know that $\pi(\lambda|\mathbf{x})$ is gamma distributed, we can also compute the predictive probability $\int_0^\infty L(\lambda)\pi(\lambda)d\lambda$, which gives the probability of observing the data over all values of λ and which becomes important later on when we describe the MCMC sampler used to avoid having to generate all possible partitions. The predictive probability is computed by rewriting $L(\lambda)\pi(\lambda)$ as

$$\frac{\frac{(\sum_{i=1}^N x_i + \phi)^{N+\alpha}}{\Gamma(N+\alpha)}}{\frac{(\sum_{i=1}^N x_i + \phi)^{N+\alpha}}{\Gamma(N+\alpha)}} \frac{\phi^\alpha}{\Gamma(\alpha)} \lambda^{N+\alpha-1} e^{-\lambda(\sum_i x_i+\phi)},$$

noting that the first fraction is just equal to one, so that:

$$\int_0^\infty L(\lambda)\pi(\lambda)d\lambda = \frac{\Gamma(N+\alpha)\phi^\alpha}{(\sum_{i=1}^N x_i+\phi)^{N+\alpha}\Gamma(\alpha)} \int_0^\infty \frac{(\sum_{i=1}^N x_i+\phi)^{N+\alpha}}{\Gamma(N+\alpha)} \lambda^{N+\alpha-1} e^{-\lambda(\sum_i x_i+\phi)}$$
$$= \frac{\Gamma(N+\alpha)\phi^\alpha}{(\sum_{i=1}^N x_i+\phi)^{N+\alpha}\Gamma(\alpha)} \times 1,$$

where the expression being integrated is the pdf of a gamma$(N+\alpha, \sum_i x_i+\phi)$ distribution and therefore evaluates to one.

Similar within-block calculations as those described above can be performed in the Bayesian formulation. For example, returning to our first example where we know a change-point occurs at $t = 3$, suppose a prior for λ of gamma(3,0.5). The posterior in block 1, $\pi(\lambda_{(1)}|\rho)$, is gamma(3+ $3, 52+0.5$), and the Bayes estimate is therefore $\hat{\lambda}_{(1)}|\rho = 6/52.5$. Similarly $\pi(\lambda_{(2)}|\rho) \sim$ gamma(2+ $3, 3+0.5$) and the Bayes estimate is $\hat{\lambda}_{(2)}|\rho = 5/3.5$. Note how both of these are pulled towards the prior mean of $3/0.5 = 6$. The Bayes estimates $\hat{\lambda}_{(2)}|\rho$ and posteriors $\pi(\lambda|\rho)$ can replace the MLEs and likelihood terms $L(\lambda|\rho)$ in Tables 3.2 and 3.3 and the rest of the calculations carried out as before. Furthermore, although we do not need it now, the predictive probability in block 1, the joint probability of observing $x_1 = 20$, $x_2 = 30$, $x_3 = 2$, integrated over all possible values of λ, can be calculated as $\Gamma(6)0.5^3/(52.5^6\Gamma(3)) = 3.6 \times 10^{-10}$.

A Gibbs sampler

In practice, rather than generating all partitions an MCMC procedure is used to simulate partitions so that these converge in probability to the posterior distribution $\pi(\rho|\lambda)$. The simulation works by setting up a vector $\mathbf{U} = \{U_1, U_2, \ldots, U_N\}$, with entry $U_t = 1$ denoting a change-point at time t and $U_t = 0$ indicating no change, and then iterates repeatedly along $\mathbf{U}$.

Suppose the current time is t, and let the index of the most recent change-point before t be denoted by a and after t by b. If there is no change-point earlier than t then $a = 0$, and if there is no change-point later than t then $b = N$.

The basic idea implemented by the simulation is to set $U_t = 0$ if the probability that the data was generated from a single block $\{X_{a+1}, \ldots, X_b\}$ is greater than that of it being generated from two blocks $\{X_{a+1}, \ldots, X_t\}$ and $\{X_{t+1}, \ldots, X_b\}$. Importantly, we can ignore any parts of the sequence before X_{a+1} and after X_b, since these are common to both. For convenience we introduce some new notation here, referring to the sequence $\{X_{a+1}, \ldots, X_b\}$ as $\mathbf{X}_{[ab]}$.

For illustration purposes, it will be useful to consider the ratio of the probabilities i.e. $\Pr(U_t = 0)/\Pr(U_t = 1)$. This ratio is a product of two further ratios, the ratio of the predictive probability of observing the data from a single block to the predictive probability of observing the data from two blocks; and the ratio of the prior probability of there being one block (between X_a and X_b) to the prior probability of there being two blocks.

The first of these ratios can easily be calculated from the formulas for predictive probability derived above. The second ratio is also easily calculated by observing that the "one-block" formulation has one fewer change-point (and one more non-change-point) than the "two-block" formulation, and that the ratios of their priors is therefore $(1-p)/p$. Thus

$$\mathcal{U}_t = \frac{\Pr(U_t = 0)}{\Pr(U_t = 1)} = \frac{m_{[ab]}(\mathbf{X}_{[ab]})}{m_{[at]}(\mathbf{X}_{[at]})m_{[tb]}(\mathbf{X}_{[tb]})}\frac{1-p}{p}$$

where $m_{[ab]}(\mathbf{X}_{[ab]})$ is the predictive probability associated with $\mathbf{X}_{[ab]}$.

The sampler moves along U_t sequentially, at each time-point generating a random number $u \sim U[0,1]$, evaluating the ratio above and allocating a change-point if $u > \mathcal{U}_t/(1+\mathcal{U}_t)$.

The final step: a prior on p

The ratio $\mathcal{U}_t$ above is as for Barry and Hartigan (1993) except that we assumed a known probability for p, the probability that any time-point is a change-point. If we want to put a prior distribution on p, a natural choice is the beta distribution, say with parameters a and b. The prior probability of a partition, previously (for fixed p) equal to $\pi(\rho) = p^{B-1}(1-p)^{N-B}$, is now given by $\pi(\rho) = \int_0^1 p^{B-1}(1-p)^{N-B}\pi(p)dp$ where, from the pdf of a beta distribution, $\pi(p) = \dfrac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}p^{a-1}(1-p)^{b-1}$. In a similar way to what we did for λ, we can show that

$$\begin{aligned}
\pi(p) &= \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\int_0^1 p^{a+B-2}(1-p)^{b+N-B-1}dp \\
&= \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\frac{1}{k}\int_0^1 kp^{a+B-2}(1-p)^{b+N-B-1}dp \\
&= \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\frac{1}{k},
\end{aligned}$$

where we define $k = \Gamma(a+b+N-1)/(\Gamma(a+B-1)\Gamma(b+N-B))$ so that the term being integrated becomes the pdf of a beta$(a+B-1, b+N-B)$ distribution.

Returning to the Gibbs sampler and the ratio of the prior distribution for one block between X_a and X_b to the prior for two blocks, it is clear that any terms in $\pi(p)$ that involve only a, b, or N will cancel, leaving only those terms involving B, the number of blocks. It is also clear that the two-block solution will end up with one more block than the one block solution. Therefore, the ratio of priors is given by:

$$\begin{aligned}
\frac{\pi(p|\mathbf{X}_{[ab]} \in \rho)}{\pi(p|\mathbf{X}_{[at]}, \mathbf{X}_{[tb]} \in \rho)} &= \frac{\Gamma(a+B-1)\Gamma(b+N-B)}{\Gamma(a+B)\Gamma(b+N-B-1)} \\
&= \frac{(a+B-2)!(b+N-B-1)!}{(a+B-1)!(b+N-B-2)!} \\
&= \frac{b+N-B-1}{a+B-1},
\end{aligned}$$

which can be substituted directly into $\mathcal{U}_t$ above.

3.3 Illustrating the Behaviour of the Change-point Model on Simulated Data

The purpose of this section is to assess the impact of different factors to the model results, in particular we want to check how sensitive the model is to:

- Length of series - how long a sequence needs to be before the model can pick up a change point
- Position of the change point - where in the sequence a change point needs to be for it to be picked up by the model
- Magnitude of the change point - how big a change point needs to be for it to be picked up by the model
- Robustness to a single noisy value - how does the model treat a big once off change between two purchases in a sequence
- Multiple change points - is the model able to pick up multiple change points in a sequence
- Different priors - do different priors cause the model to produce different results

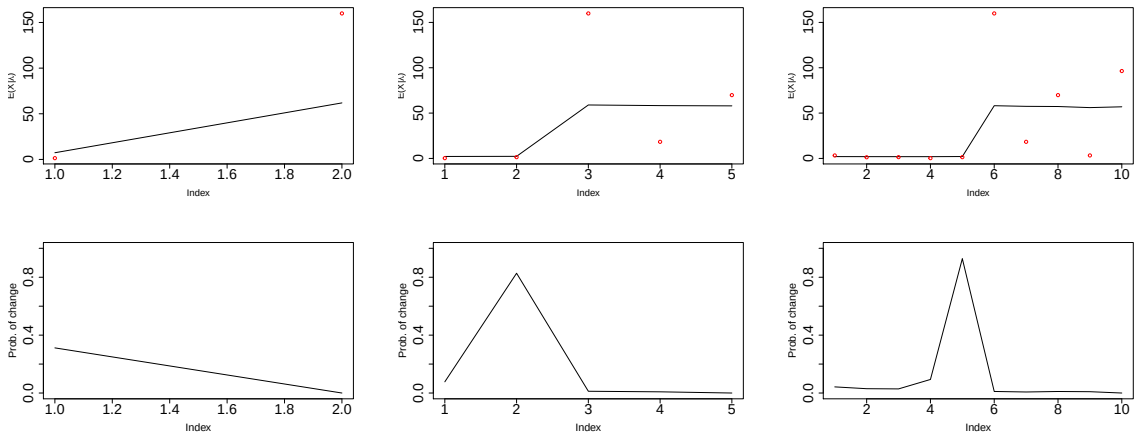
For the purpose of clear demonstration, this illustration will be carried out using simulated data as with this type of data we will know exactly whether the change point did occur, and where/when (i.e. the position).

3.3.1 Sensitivity of Results to the Length of the Time Series

To assess this factor (i.e. checking the minimum number of observations the model requires in order to be able to pick up the change point), we use a sequence which we know has a change point, but just taking a few observations around the change point (before and after, with the change point being somewhere in the middle). The purpose of this exercise is to determine the number of observations the model requires in order to be able to reliably pick up the change point.

The results shown in Figures 3.1a, 3.1b and 3.1c were performed on a dataset that was generated from an exponential distribution with parameter λ initially set to 2 at time $t = 1$, then changing to 50 at times $t = 2$, $t = 3$ and $t = 6$ in Figures 3.1a (with 2 observations), 3.1b (with 5 observations) and 3.1c (with 10 observations), respectively. The results in Figures 3.1a, 3.1b and 3.1c show that the model needed at least 10 observations to pick up the change point reliably, as nothing clearly showed when we used 2 observations, but signs of the change point

were already visible when we used 5 observations. This would be quite encouraging if this were real purchase data, as after only 5 purchases, we are already able to detect a change within only a few more. But since this is just one simulated dataset, we cannot conclude that this is how the model will perform in general, as this could be just how it worked for this dataset.



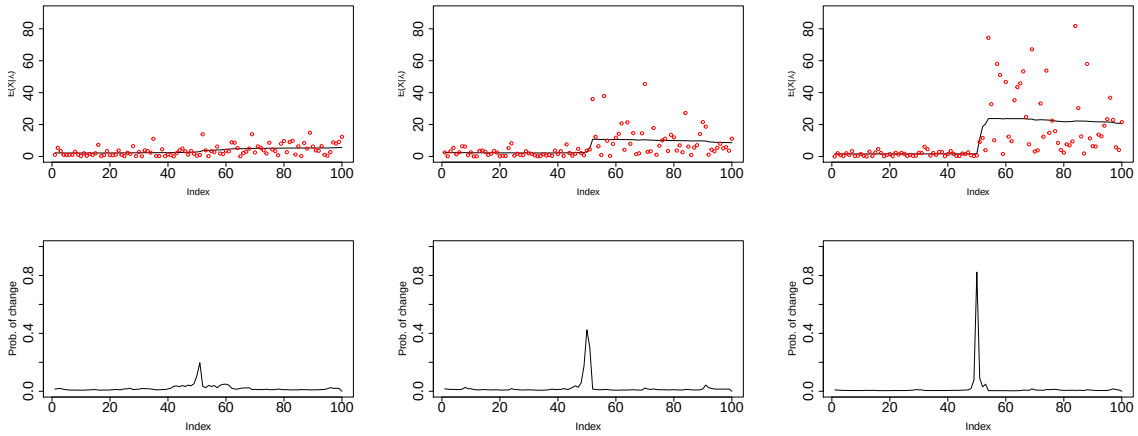
(a) 2 observations (b) 5 observations (c) 10 observations

Figure 3.1: Change-point results generated from an exponential distribution, with λ initially set to 2, then changing to 50 at time $t = 2$, $t = 3$ and $t = 6$ in Figures 3.1a, 3.1b and 3.1c respectively. The red dots in the top graph represent data points, while the black line represents the posterior expected inter-purchase time obtained from the change-point model. The black line in the bottom graph represents the probability of a change point.

3.3.2 Sensitivity of Results to the Magnitude of the Change

From the results shown in Figures 3.1a, 3.1b and 3.1c we can see that, the more the observations, the better job the model does at picking up the change point. In addition to this, we now want to find out how big a change should be before the model can pick it up. In other words, we would like to see if the model is as sensitive to small changes as is to bigger ones. To evaluate this we look at a sequence that has a change point occurring at the same time, but varying in the magnitude of the change.

The results shown in Figures 3.2a, 3.2b and 3.2c were performed on a dataset that was generated from an exponential distribution with 100 observations, and parameter λ initially set to 2, then changing at time $t = 50$, to 10, 20 and 50 in Figures 3.2a, 3.2b and 3.2c, respectively. The results for this exploration are shown in Figures 3.2a, 3.2b and 3.2c, and it is evident from these results that the smaller the change, the harder it is for the model to pick it up. The model struggles to pick up the change point if the change is very small, but improves as the change gets bigger.



(a) Final $\lambda = 5$ (b) Final $\lambda = 10$ (c) Final $\lambda = 20$

Figure 3.2: Change-point results generated from an exponential distribution with 100 observations, and λ initially set to 2, then changing at time $t = 50$. The red dots in the top graph represent data points, while the black line represents the posterior expected inter-purchase time obtained from the change-point model. The black line in the bottom graph represents the probability of a change point.

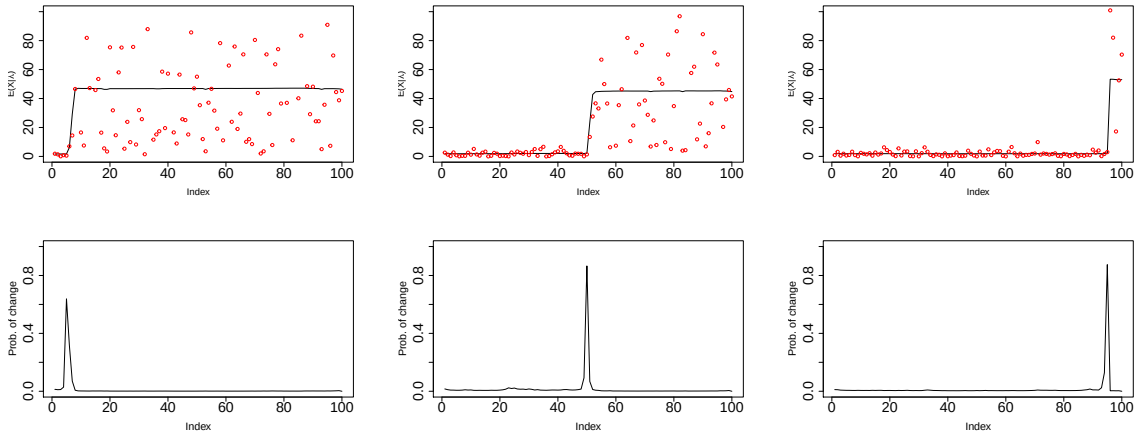
3.3.3 Sensitivity of Results to the Position of Change-Point

Having established above that the model needs a longer sequence, and a significant change to be more accurate, we now want to check if the position of the change point has any effect on the model at all. Provided that the sequence is long enough, and that the change is sizable, we want to see whether the model can pick up the change point wherever it occurs in the sequence, (i.e. at the beginning, middle, or end of the sequence).

The results shown in Figures 3.3a, 3.3b and 3.3c were performed on a dataset that was generated from an exponential distribution with 100 observations, and parameter λ initially set to 2, then changing to 50 at times $t = 6$, $t = 50$ and $t = 96$ in Figures 3.3a, 3.3b and 3.3c, respectively. Figure 3.3a shows that even though the change point occurs very early (at $t = 6$) in the purchase history, the model does a very good job at picking it up. Likewise, the results are sensitive in the middle, and as sensitive at the end of the series, see Figure 3.3b and Figure 3.3c, respectively.

3.3.4 Robustness of the Results to a Single Noisy Value

We already have established that the model easily picks up large change points, but what if for an instance we have a case whereby we did not necessarily have a change point, but just a considerable change between two purchases. That is, there still was a significant change from a previous purchase to the next, but that was just for that particular purchase, and the trend continued normally after that abnormal change (spike). We therefore want to assess how the model results are sensitive in such a scenario.



(a) Change at time $t = 6$ (b) Change at time $t = 50$ (c) Change at time $t = 96$

Figure 3.3: Change-point results generated from an exponential distribution with 100 observations, and λ initially set to 2, then changing to 50. The red dots in the top graph represent data points, while the black line represents the posterior expected inter-purchase time obtained from the change-point model. The black line in the bottom graph represents the probability of a change point.

Figure 3.4 shows a series generated from an exponential distribution with parameter $\lambda = 2$, with the exception of a single point at $t = 50$ (where $\lambda = 50$). Results show that the change-point model identifies the point at $t = 50$ as a change-point, even though the "change" only persists for a single time period. This is because the generated value at $t = 50$ is extremely unlikely to have been generated from the same distribution as the rest of the series.

3.3.5 Sensitivity of Results to Multiple Change Points

Previous sections have been restricted to results on a single change-point. In this section we examine the ability of the change-point model to detect more than one change-point. We generated data from an exponential distribution with 100 observations, and parameter λ initially set to 2, then changed to 50 at time $t = 25$, then back to 2, at time $t = 50$, and again to 50, at time $t = 75$. In Figure 3.5 the results show that if there exist more than 1 change point in the sequence, the model is able pick up all of them. There were three change points in the sequence, and all were detected by the model.

3.3.6 Sensitivity of Results to Different Priors

For the final point of our demonstration, we assess the sensitivity of the results to different priors. Again, we did this by using a sequence that we know is long enough, and the change is big enough for the model to pick up, but just experimented with different priors as shown in Figure 3.6. From what we see, priors seem not to have a significant effect on the model results, as the results show no significant changes for different priors, despite the prior changes being

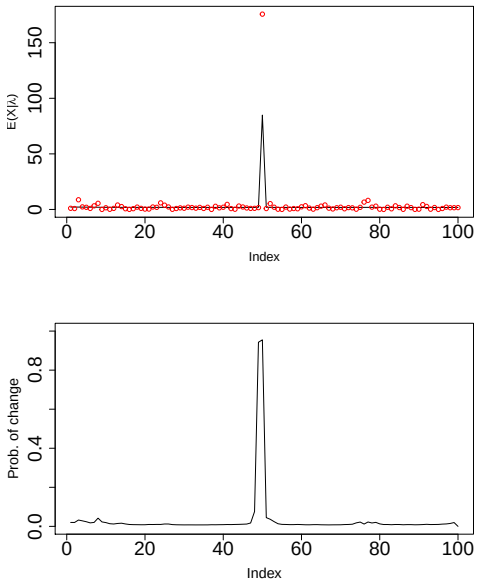


Figure 3.4: Change-point results generated from an exponential distribution with 100 observations, and $\lambda = 2$, with the exception of the single point at $t = 50$ (where $\lambda = 50$). The red dots in the top graph represent data points, while the black line represents the posterior expected inter-purchase time obtained from the change-point model. The black line in the bottom graph represents the probability of a change point.

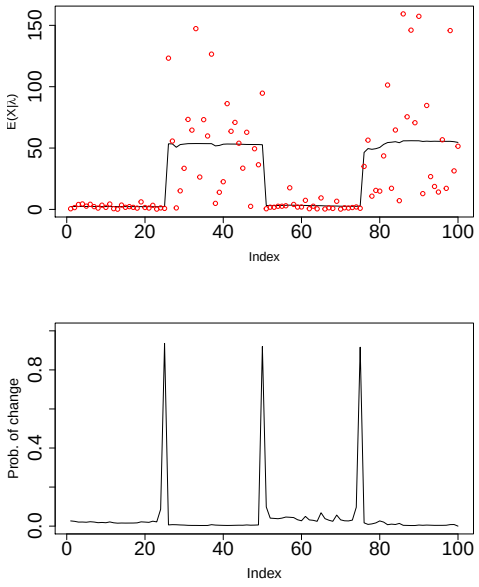


Figure 3.5: Change-point results generated from an exponential distribution with 100 observations, and λ initially set to 2, then changed to 50 at time $t = 25$, then back to 2 at time $t = 50$, and again to 50 at time $t = 75$. The red dots in the top graph represent data points, while the black line represents the posterior expected inter-purchase time obtained from the change-point model. The black line in the bottom graph represents the probability of a change point.

quite big (in the λs).

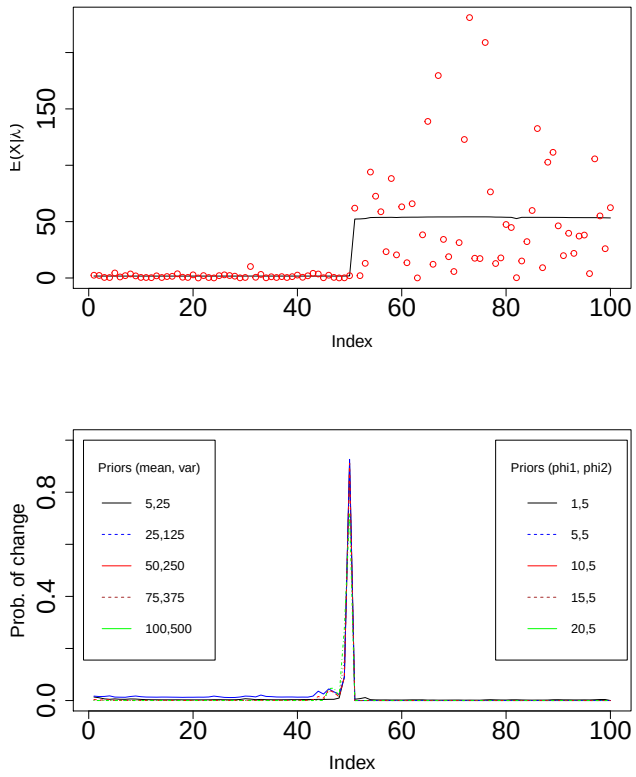


Figure 3.6: Change-point results fitted using different priors, generated from an exponential distribution with 100 observations, and λ initially set to 2, then changing to 50 at time $t = 50$. The red dots in the top graph represent data points, while the black line represents the posterior expected inter-purchase time obtained from the change-point model. The black line in the bottom graph represents the probability of a change point.

3.4 Model Implementation

The data we used in this chapter consists of 2000 customers' purchase histories (for both stores and brand models), these customers being randomly sampled from a larger set of customers, such that they have at least 50 purchases for stores (since store sequences of purchases are relatively longer), and 20 purchases for brands (since brand purchases generally result to shorter sequences). For both models there will be no limit on the maximum number of observations.

Various kinds of inter-purchase times can be extracted from a panel dataset such as the one we analyze in this chapter. For example, for a particular customer, times between:

- any purchase (of any product, at any store),
- a store visit (purchases of any product, at a particular store),

- a product purchase (purchases of a particular product, at any store),
- purchases of a particular product at a particular store

Bayesian change-points are computationally intensive, and with 144 stores and over 3000 different products it is not feasible to run analyses on each store and product. We therefore chose to fit a Bayesian change-point model to each of two inter-purchase time datasets:

- For each of 2000 customers, we recorded the dates on which they visited any store (i.e. made any purchases at all). Data for each customer consists of a time series whose length is equal to one less than the number of store visits they made, and whose entries are the number of days between consecutive store visits. The sample of 2000 customers was selected from a larger pool of customers, with the requirement that selected customers had to have more than 50 store visits. This ensures that the change-point models have sufficiently long time series to work with.
- For each of 2000 customers, we recorded the dates on which they purchased a particular brand of interest. Data for each customer consists of a time series whose length is equal to the one less than the number of times they purchased this brand, and whose entries are the number of days between consecutive purchases of the brand. The sample of 2000 customers was selected from a larger pool of customers, with the requirement that selected customers had to have bought the brand more than 20 times. We chose the brand that had the largest number of purchases across the entire dataset as the brand of interest.

The prior probability (on p) we use in our models is 0.05; this was chosen to be this small in order to ensure that the chances of there being a change point in the sequences is very small. This increases the likelihood of the points the model detects as potential change points to be actually change points. The prior mean chosen on the expected inter-purchase times (λ) is 4, this is because the actual inter-purchase times mean on the data we used is 6, so we chose this prior to be slightly less than the actual mean. The Bayesian change-point model and the MCMC sampler used to fit the model were coded in R (version 3.5.2). To run the analysis, we used 4 x t2.medium instances on Amazon Web Services EC2 (Elastic Cloud computing services). We allocated batches of 50-100 sequences to each instance.

3.5 Illustrating the Use of the Change-point Model in Practice

In practice, the Bayesian change-point (BCP) model would be used to analyze individual customer's purchase histories, and, if a change is detected, to intervene in some way (i.e. send them a special offer coupon, etc). So in this section we do a similar illustration to the previous section, but focusing on practical interpretation and illustrating some of the patterns we found in the dataset. The results were run for both change point models, i.e. store and brand

customer inter-purchase times models, and we will show the demonstrations using results from both models.

3.5.1 Sensitivity of the Results to the Magnitude of the Change-point

Figure 3.7a shows that at the beginning of the sequence, the customer had a long waiting time between store purchases, waiting for up to 80 days before visiting the store again , with the inter-purchase times between the third and forth purchases being 65 days. However, they seemed to have shopped more often from the fourth purchase onwards (every 5 days on average). Since we are not sure of the purchase behavior before the beginning of the analyzed sequence (i.e. whether this long wait persisted for only these 3 purchases), we cannot really tell what could have been the reason for this change. However, this is still quite a big difference in inter-purchase times, one that the store manager should investigate to ensure that it spreads out to other customers as it is a positive change (i.e. inter-purchase times became shorter) for the store.

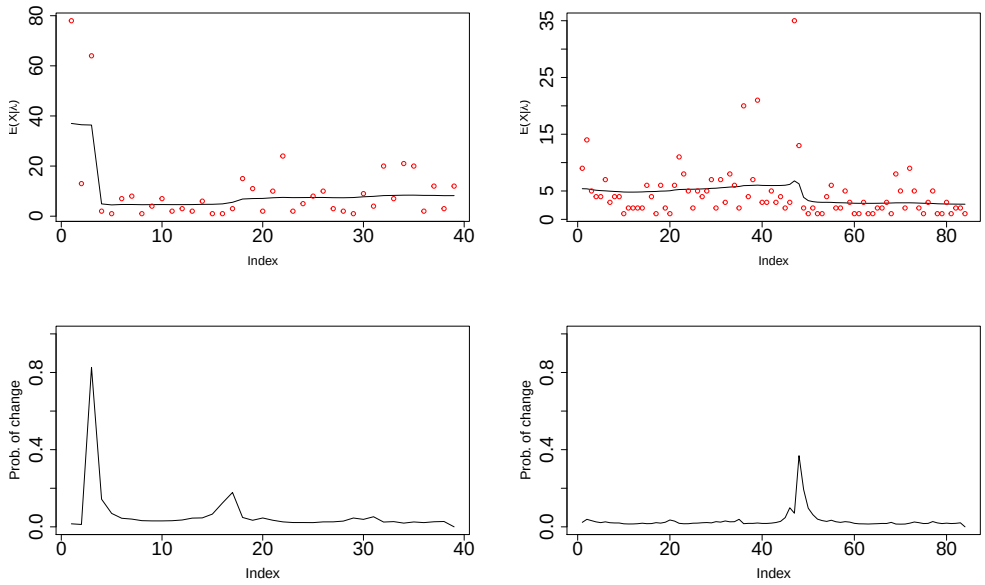
A similar result of a positive change for a brand purchase is shown in Figure 3.7b. The change is however not as large as the store change shown in Figure 3.7a, as the actual change is just 2 days. However, given that this was a brand purchase, 2 days can make a lot of difference to the brand's share of wallet than it would to the store one, and thus making acting on this change as vital.

3.5.2 Sensitivity of the Results to the Length of the sequence

Figures 3.8a and 3.8b show store results of the same sequence, but with different sequence lengths. In Figure 3.8a the sequence has close to 200 observations, compared to the 10 observations in Figure 3.8b, but the change-point is as evident in the shorter sequence as it is in the longer one. These results are quite encouraging, as the model does not require a long purchase history to reliably detect change points, and this is quite useful as changes can be detected as early as possible. This can allow managers to act on time before incurring loses or losing out on growth opportunities.

3.5.3 Robustness of the Results to a Single Noisy Value

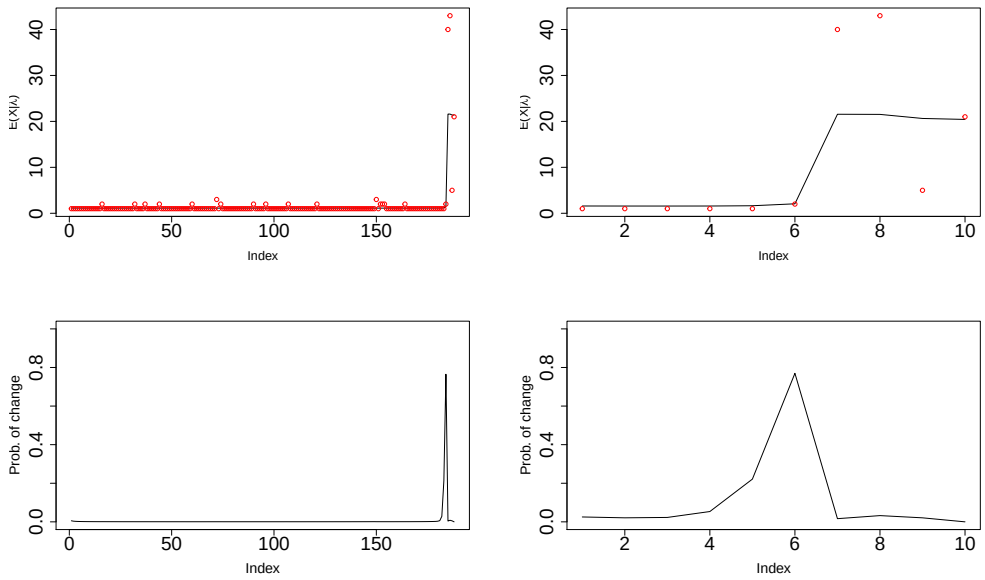
In practice, the general rule is that all spikes need to be investigated until their root cause is known and understood, then decisions on whether to act on these spikes be taken afterwards. The results in Figure 3.9 show a classical example of a spike that could be experienced in practice, whereby a huge change persist for only a single purchase. The model however identifies this spike (point at $t = 48$) as a change point, despite the change persisting only for a single purchase compared to the results in the above sections where changes persist for a few purchases. Again, after further investigations a store manager would have to decide on whether to treat this point as a change point and act on it or just consider it a random occurrence (i.e. maybe the customer was away for these 6 months) and do nothing about it.



(a) Obvious change

(b) Subtle change

Figure 3.7: Change-point results from actual data, with Figure 3.7a showing an instance where the change is obvious, and 3.7b where it is subtle. The red dots in the top graph represent data points, while the black line represents the posterior expected inter-purchase time obtained from the change-point model. The black line in the bottom graph represents the probability of a change point.



(a) Whole sequence

(b) Few observations

Figure 3.8: Change-point results from actual data, with Figure 3.8a showing an instance where the full dataset is observed, and 3.8b where only a sample of the data is used. The red dots in the top graph represent data points, while the black line represents the posterior expected inter-purchase time obtained from the change-point model. The black line in the bottom graph represents the probability of a change point.

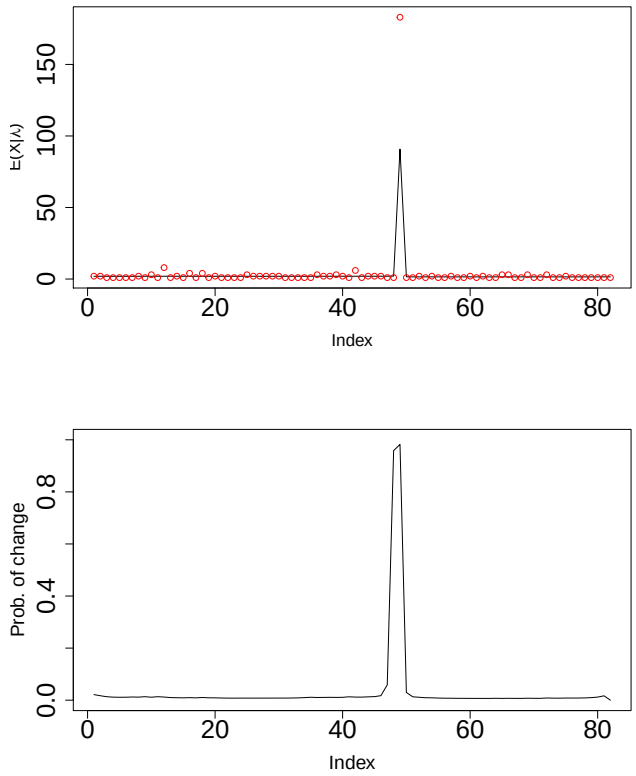
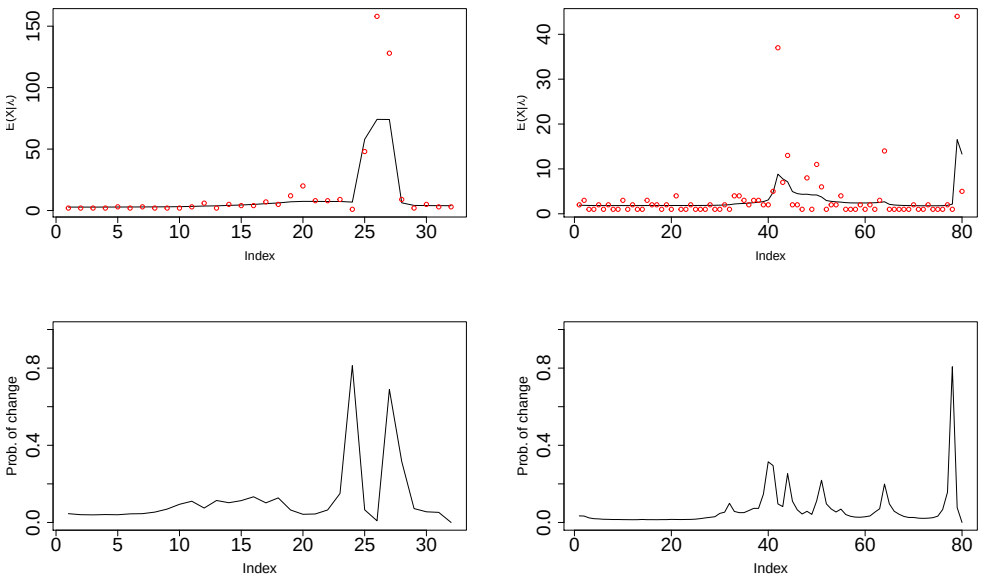


Figure 3.9: Change-point results from actual data with 82 observations, and λ changing from 1 to 100 at $t = 48$, then again to 1 at $t = 49$. The red dots in the top graph represent data points, while the black line represents the posterior expected inter-purchase time obtained from the change-point model. The black line in the bottom graph represents the probability of a change point.

3.5.4 Robustness of the Results to Multiple Change Points

Figures 3.10a and 3.10b show that it is possible for customers to experience multiple change-points in their purchase history. These multiple changes can be obvious as we can see in Figure 3.10a, or subtle as shown in Figure 3.10b. In Figure 3.10a, the customer seem to buy often (almost every day), then less often for a few purchases (2, 4 and 5 months between the following 3 purchases), then go back to often again (daily purchases). This could suggest that the customer was away for that particular year, and only shopping when visiting home, till they moved back home again and started shopping on a daily basis again. On the other hand, the changes in Figure 3.10b are, firstly not clear, and secondly quite frequent. The results show 4 potential (unclear) change-points between purchases at $t = 40$ and $t = 64$, and a clear change-point at $t = 79$. The brand manager would need to do two things here; firstly decide on whether to treat that patch on the middle as change-points, and if yes investigate why this customer changes their behavior this often, and perhaps incentivize them as they pose a risk of leaving; secondly, the manager can disregard the middle purchases and concentrate on the change-point at $t = 79$ which is clear and large, and decide how to handle it (it is not clear though how long this persisted as it occurred at the end of the sequence).



(a) Clear multiple

(b) Subtle multiple

Figure 3.10: Change-point results from actual data, with Figure 3.10a showing clear multiple change points, and 3.10b showing unclear change points. The red dots in the top graph represent data points, while the black line represents the posterior expected inter-purchase time obtained from the change-point model. The black line in the bottom graph represents the probability of a change point.

3.6 Results and Discussion

In this chapter we ran two change-point models (for store and brand) on full samples of 2000 customers and here we present aggregated results for both models. The objective is to assess how often people change their behaviour over the course of a year, where this behaviour is measured by the amount of time between their store visits or product purchases. To determine the nature of the change (i.e. whether the change is favorable or unfavorable), the user would need to inspect the series (as illustrated in section 3.5) to see if it is an increase or decrease. Tables 3.4 and 3.5 contain results for store and brand models, respectively. The results show the proportion of customer's purchase histories with no change points, single change point, and multiple change points. Tables 3.6 and 3.7 show (for store and brand level results, respectively) the proportion of sequences for which the differences between maximum and minimum estimated inter-purchase times (R) equals or exceeds 1, 2 and 5, respectively. R Band on the other hand shows proportions of sequences that had a certain number of change points (i.e. ¡1, 1, 2-4 or 5+). For example, R on the first row in Table 3.6 shows that 70% of sequences had at least 1 change point, whereas R Band shows that 34% of these sequences had just one change-point, 25% had between 2 to 4 change-points, and 11% had more than 4 change-points. The results in the R column give three points of the empirical cumulative distribution of λ_t. Note that the ranges relate to the underlying parameter of λ_t and not the probability that there is a change-point at time t.

Table 3.4: Number of Change Points - Store Inter-Purchase Times

Length	Count	CPs = 0	CPs = 1	CPs = 2	CPs = 3+
50 - 59	278	44.3%	44.3%	8.9%	2.5%
60 - 69	294	42.2%	45.3%	9.5%	3%
70 - 79	226	30.7%	57.9%	6.6%	4.8%
80 - 89	197	22.6%	60.3%	11.1%	6%
90 - 99	176	13.5%	74.2%	10.1%	2.2%
100 - 109	162	18.3%	70.7%	7.9%	3%
110 - 119	125	22.8%	66.1%	8.7%	2.4%
120 - 129	122	28.8%	58.4%	10.4%	2.4%
130 - 139	90	21.7%	64.1%	9.8%	4.3%
140 - 149	96	29.6%	57.1%	10.2%	3.1%
150 - 159	73	42.5%	49.3%	6.8%	1.4%
160 - 169	46	41.3%	32.6%	15.2%	10.9%
170 -179	48	56.3%	35.4%	6.3%	2.1%
180 - 189	32	53.1%	46.9%	0%	0%
190+	35	65.7%	20%	8.6%	5.7%
AVG	2021	32.1%	55.4%	9%	3.5%

The store results in Table 3.4 show that, contrary to what one might expect, that longer series means more chance for change, it is a series of intermediate length that has the largest probability

Table 3.5: Number of Change Points - Brand Inter-Purchase Times

Length	Count	CPs = 0	CPs = 1	CPs = 2	CPs = 3+
19 - 29	738	81.1%	17.6%	1.4%	0%
30 - 39	509	75.9%	20.9%	2.5%	0.6%
40 - 49	302	67.3%	28.4%	3%	1.3%
50 - 59	204	53.2%	42.4%	4.4%	0%
60 - 69	111	44.6%	51.8%	3.6%	0%
70 - 79	59	20%	65%	15%	0%
80 - 89	33	24.2%	66.7%	6.1%	3%
90 - 99	20	25%	70%	5%	0%
100+	24	33.3%	54.2%	4.2%	8.3%
AVG	2008	68.9%	27.7%	2.9%	0.5%

Table 3.6: Proportion of Store Sequences in which the Difference Between Maximum and Minimum Estimated Inter-Purchase Times Exceeds R

	R Band				R		
Length	< 1	$=1$	2-4	5+	$>= 1$	$>= 2$	$>= 5$
50-59	30	34	25	11	70	36	11
60-69	39	36	17	8	61	25	8
70-79	46	32	13	9	54	22	9
80-89	38	41	13	9	62	22	9
90-99	52	29	14	5	48	19	5
100-109	52	35	9	4	48	13	4
110-119	50	39	8	3	50	11	3
120-129	59	30	7	3	41	10	3
130-139	63	24	8	5	37	13	5
140-149	67	21	6	5	33	11	5
150-159	70	23	7	0	30	7	0
160-169	48	26	24	2	52	26	2
170-179	71	21	6	2	29	8	2
180-189	56	41	0	3	44	3	3
190+	69	20	6	6	31	11	6
Grand Total	48	32	13	6	52	20	6

Table 3.7: Proportion of Brand Sequences in which the Difference Between Maximum and Minimum Estimated Inter-Purchase Times Exceeds R

	R Band				R		
Length	< 1	$=1$	2-4	5+	$>= 1$	$>= 2$	$>= 5$
30-39	15	36	36	13	85	50	13
40-49	20	42	27	10	80	37	10
50-59	28	45	20	7	72	27	7
60-69	38	42	17	3	62	20	3
70-79	38	45	5	12	62	17	12
80-89	52	33	9	6	48	15	6
90-99	65	25	10	0	35	10	0
100+	58	25	17	0	42	17	0
Grand Total	18	35	31	16	82	47	16

of one or more change points. Relatively long series and relatively short series both have higher chances of no change. It is hard to say why this could be the case, one possibility though would be that customers start by being loyal to their preferred store, then move onto trying other different stores, and after experimenting with those other different stores return to their preferred store and stick with it. Also, people who shop often are very regular, probably because they are shopping for a family, or for work, and times between purchases in such instances tend to be constant.

The brand results presented in Table 3.5 show that on average, people showed fewer changes in their inter-purchases times for this brand exhibit than their inter-purchase times for stores. This however seems to be largely because store-level sequences are longer than brand-level sequences, rather than due to any fundamental difference in behaviour. To investigate this further, we compared store-level and brand-level results of the same length, this involved taking sequences with lengths in the range 50-99 (i.e. 50-59, 60-69, 70-79, 80-89 and 90-99) for both brand-level and store-level and comparing their results. We saw that 33% of sequences for brand have no change point, compared to 31% for store. These proportions for sequences that have at least one change point are 59% and 56% for brand and store, respectively. As we can see, these numbers are very similar, suggesting that the substantial difference we saw in results between store and brand could very well be due to the fact the sequences lengths analyzed are different. Also, for brand-level results, the longer the sequences, the higher the chance of change (no intermediate maximum, as observed for store-level results).

Next, we look at Tables 3.6 and 3.7, which contain proportion of customers for which the range between maximum and minimum inter-purchase times is greater or equal to some cut-off, R. The ranges give some idea of the changes in estimated inter-purchase times that occur from sequence to sequence. The majority of inter-purchase times change very little as 94% and 84% of customers (averaged over all sequence lengths) change their inter-purchase times by less than

5 days for store-level and brand-level customers, respectively. This is good for management, as it suggests that even though the change in inter-purchase times for some customers can be huge (as seen in Section 4), most customers experience very small changes, hence most average inter-purchase times changes are less than 5. This also was expected as the average inter-purchase times in our data is 4 and 10 days for store-level and brand-level customers, respectively (and 6 overall), hence there are more customers with estimated inter-purchase times greater than 4 in the brand-level results compared to the store-level.

In concluding, majority of customers for store inter-purchase times experience at least one change point in their purchase behaviour, with 1 out of 5 of these having multiple change points. Also, for brands, a handful of sequences have at least a single change point, and out of those sequences having a change point, 11% go on to show multiple change points. Crucially, this means the change-point model can be a useful tool in management if most people do change their behaviour, then perhaps there are some things that managers can do to take advantage of this information (i.e. intervene by sending offer coupons). These results are somewhat different to those of brand-selection obtained by Clark and Durbach (2014), as they concluded that only a minority of customers (5-25%) showed changes in their purchase behaviours. Interestingly though, when we compare same sequence lengths (20-39) as those analyzed by Clark and Durbach (2014), the results look similar, as 19% (compared to the 5-25%) of sequences in these bands contained at least one change point.

3.7 Summary and Recommendations

The objective of this chapter was to assess how often people change their behaviour over the course of a year, where this behaviour is measured by the amount of time between their store visits or product purchases. This involved developing two change-point models, one focusing on modeling time between store purchases, and the other on modeling time between brand purchases. This chapter started with an introduction section, which covered problem statement, overview of the chapter, and previous work discussion. In the following section we then discussed model description, data and tool of analysis, before concluding with discussing model implementation. In Section 3, we did a demonstration of the change-point model, with the main aim being to illustrate the output of the model, and to explore (using some simple simulated sequences) how the change-point model responds to changes in some key inputs; these included sequence length, size of change, position of change-point, multiple change-points, various priors, and single noisy value. The way the change-point model responded confirmed some common-sense intuition (i.e. model works better with relatively longer series than relatively shorter ones, the model easily picks up big changes than small ones etc). In the following section we did a similar demonstration, but this time illustrating the use of the change-point model in practice, where we focused on practical interpretation and illustrated some of the patterns we found in

the dataset. The change-point model performed well on the actual data, picking up similar patterns it picked up on the simulation data, and thus showing robustness and consistency.

On the actual results, the main message was that majority of sequences for stores (68%) had change point(s), and this number was less than half for brand, being (31%). Proportion of sequences with multiple change points was 13% and 3% for stores and brand, respectively. It was also discussed that though the proportion of sequences with change points seemed to be more for store than that of brands, this did not necessarily mean that there are more changes in store inter-purchase times than those of brand. But this was rather due to the different sequence length sizes analyzed for store and brands. On the other hand, the results seemed to be different to those obtained by Clark and Durbach (2014), but analyzing a portion of sequences of same lengths as those analyzed by Clark and Durbach (2014), lead to similar results.

Chapter 4

Using Deep Learning Methods to Predict Repeat Customers

4.1 Introduction

4.1.1 Background to the problem

It is vital for retailers to increase sales growth, and there are various possible ways in which they can achieve this. Acquiring new customers (whilst retaining current ones) is one of the standard strategies. Acquiring new customers involves contacting (through marketing campaigns) potential customers not in the database, and try to sell to them. On the other hand, retaining existing customers requires constant customer engagement to ensure that customers are satisfied with your services all the time. The other strategy of increasing profit is through cross-selling and up-selling. Both cross-selling and up-selling involve marketing only to already existing customers. Cross-selling is a strategy of providing the opportunity to purchase additional items (different to what they already buy) offered by the seller to the existing customers. This often involves offering the customer items that compliment (in some manner) the original purchase. On the other hand, up-selling is whereby a customer is persuaded to make an additional purchase of the same product or purchase a more expensive version of the product (Kubiak and Weichbroth, 2010).

Customer-Relation Management (CRM) which is an approach to manage a company's interaction with current and potential customers is at the center of these strategies. Buttle and Francis (2006) describe CRM as an "integrated information system that is used to plan, schedule and control the pre-sales and post-sales activities in an organization. CRM embraces all aspects of dealing with prospects and customers, including the call centre, sales force, marketing, technical support and field service. The primary goal of CRM is to improve long-term growth and profitability through a better understanding of customer behaviour. CRM aims to provide more effective feedback and improved integration to better gauge the return on investment (ROI) in these areas". Most of the times, customers respond better to campaigns that have some form of incentive attached to them. For example; for an acquisition campaign, some customers may

respond better if a clothing retailer offers them a discount on their first purchase after joining the store, for a cross-sell campaign, a customer would be more interested in trying new product(s) if it was at a discounted price than at a normal price. Because of this, most campaigns of these types contain some form of incentive as means of trying to attract more customers.

Most retailers have now adapted statistics (statistical modeling, data science, data mining etc...) to use in conjunction with CRM in order to optimize their processes. The ways in which statistical modeling can be used to optimize business processes include; using propensity to buy models (that predict who is highly likely to buy) when campaigning to potential customers (Royston-Webb, 2018), building change point models to detect early unwanted changes in customer behaviour and intervene on time (Clark and Durbach, 2014), market basket analysis to predict the next best product to sell to a customer (Gangurde et al., 2005). When building these models, different attributes can be considered, these include; who the customers are (i.e. customer demographics), what they buy (brand selection/choice), how they buy (purchasing behaviour; i.e. basket size and basket value) and how often they buy (frequency of purchases). Some models can make use of all these attributes, but which attributes to use is a decision to be made by the model builder depending on the overall objective of the model.

4.1.2 Objective

This chapter covers the up-selling and cross-selling strategies mentioned above. In this chapter we build a model to predict which customers are likely to become repeat customers after being given a special offer. That is, if a customer was given an offer (say a discount) to buy a particular product, what is their likelihood of continuing to buy that product after the offer expires? The offer can be given to customers who never purchased the product before (as means of cross-selling) or to people who have previously purchased the product (up-selling campaign). This chapter looks at customers who have not necessarily purchased the product previously, but have bought in the department to which that offer belongs (so covers both up and cross selling). In other words, a customer must have either purchased the product in question before or purchased another product in the same department. Customers who have not purchased in the department before will not be considered. The predictor variables for our model are the times between any previous purchases (i.e. inter-purchase times). So in this chapter we will be looking at customers department inter-purchase times (time between purchases in the department to which the product they received an offer for belongs), and whether this allows us to predict if they will become a repeat customer after being sent a special offer.

These models will be built using deep learning methods, neural networks, focusing on convolutional neural networks (CNN) and recurrent neural networks (RNN). This is mainly because, our input data is a time series, and is sequential in nature, so CNN and RNN provide good ways of dealing with sequential inputs. We fit the CNN and RNN models, and compare them to feed-

forward neural network (FNN) and a decision tree, which use the same inputs (inter-purchase times), but do not take into account the sequence of the data (i.e. we could swap around the inter-purchase times and the results would be the same). In total we will look at four different models, the first being the CNN model, and the rest being variations of the RNN family, namely; Gated Recurrent Unit (GRU), Stacked Gated Recurrent Unit (SGRU) and Bidirectional Gated Recurrent Unit (BGRU). We will also build some traditional classification models in decision trees to use as benchmark for the deep learning models. The main reason behind this, is because as we have seen in Chapter 2, and also as shown in Makridakis et al. (2018), deep learning methods do not always outperform classical statistical methods. So we want to assess whether there is value in fitting our dataset using these complex deep learning models or traditional models would suffice.

4.1.3 Outline

This chapter comprises of 5 sections (inclusive of this one which is an introductory section). In the next section we discuss deep learning methods for sequence modeling. We start by discussing sequence modeling, followed by previous work on sequence modeling. Then next we discuss the neural network methods, followed by the discussion of model accuracy for binary models and metrics used to measure this accuracy. In particular we will discuss misclassification, specificity, sensitivity, Gini statistic, Area under the curve (AUC), F1-score and Kappa. The following section will discuss the dataset being used in this chapter, followed by the methodology, model implementation, and conclude with the softwares on which the models will be fitted. In Section 4 we present and discuss results, and conclude with recommendations. The last section summarizes the whole chapter.

4.2 Deep Learning Methods for Sequence Modeling

Sequence modeling has been growing in recent years due to its application to day-to-day business problems and practices. This is mainly because most of the data in the current world is in the form of sequences (Tavish, 2019). Examples of this type of data include; a number sequence, image pixel sequence, a video frame sequence, an audio sequence or a text sequence. Some examples of sequence modeling applications are; speech recognition to listen to the voice of customers, machine language translation from diverse source language to a more common language, topic extraction to find the main subject of customer's translated query, speech generation to have conversational ability and engage with customers in human like manner, and text summarization of a customer feedback to work on key challenges (Tavish, 2019).

Sequence models can be of various forms based on the type of input they take, and the output they generate. These include sequence generators - these models generally take scalar inputs and produce a sequence (i.e. model that takes a single word as an input and generates a sentence

as an output); sequence to sequence models, which take sequence inputs and produce sequence outputs (i.e. model that takes a sentence as an input and generates another sentence as an output) (Tavish, 2019). It is also possible for a sequence model to take a sequence input, and produce a scalar output, and in this chapter we use this type of sequence modeling as we feed a sequence of inter-purchase times into the model, and produce a binary output (i.e. continued to purchase product or did not continue) as result.

Several studies have been published on predicting future customer behaviour (Bahari and Elayidom (2015); Xu and Walton (2005); Garland and Gendall (2004)). In a study similar to ours, Lazarov and Capota (2007) used inter-purchase times (together with customer perceptions and demographic data), to predict (a binary outcome) whether a customer is likely to churn for a service provider company. The inter-purchase times were defined as time between purchases of two articles, and period between calls. The idea behind the study was to find the churners and reasons for quitting, so that rapid action could be taken by the marketing department in order to prevent churn properly. Since there is usually not enough time to address all likely churners, for customers with the highest probability of churning, it would be predicted how much revenue a service provider would get over the period of customers's stay. This would help identifying the most valuable customers and efforts of retaining would be made for these customers first. The models were fitted using Regression analysis, Decision Trees, Neural Networks and Rule based learning methods, and it was conluded that neural networks are superior in performance as opposed to other models.

In Chapter 1 we discussed basic Artificial Neural Networks (ANN), which are essentially based on human brain which can be described as a biological neural network, that is an interconnected web of neurons that transmit information in the form of electric signals, and their original development (Rosenblatt (1962); Rumelhart et al. (1986); Bishop (2006)) was an attempt to find mathematical representations of information processing in the brain. In an ANN, neurons, axons and dendrites are defined as nodes (input, hidden and output layers), arrows and inputs received by each node, respectively. It was pointed out in Chapter 1 that, if an ANN is structured in such a way that, the information flows from the input layer to the output layer, with output from any one layer acting as input to the next layer until the information reaches the output layer, it is called a Feed Forward Neural Network (FFNN).

Building on this background of the FFNN, the rest of the section will discuss, based on the work presented in various publications (Dlamini (2018); Chung et al. (2014); Husken and Stagge (2003)) the theoretical framework of two other neural network techniques in Convolutional Neural Network (CNN) and Recurrent Neural Networks (RNN). In total we will discuss four different models, the first being the CNN, and the rest being subtypes of the RNN in GRU, SGRU and BGRU.

4.2.1 Convolutional Neural Networks (CNN)

The Convolutional Neural Network (CNN) is a subtype of an FFNN, and is inspired by the visual cortex of animals, which is the part of a brain that is crucial in the processing of visual information (Hubel and Wiesel, 1962). CNNs have been widely applied and have received a lot of success in image classification and computer vision tasks due to their inspiration from the visual cortex (Hubel and Wiesel, 1962). CNNs are based on three fundamental principles; *a*) local receptive fields, *b*) shared weights, and *c*) spatial or temporal subsampling (LeCun et al. (1998a)). Table 4.1 (LeCun et al. (1998a),(Cornelisse, 2018);(Karn, 2018)) contains descriptions of CNN concepts being discussed in this section.

Local Receptive fields

The idea of a receptive field has been prevalent in neuroscience for a long time (see Hubel and Wiesel (1962)). This can be defined as the region in the input space that a particular CNN's feature is looking at. The receptive field of a neuron in a layer is the cross-section of the previous layer from which neurons provide inputs. The basic idea of a receptive field is to extract local features and then combine them to make more complex patterns. That translates to local transformations and therefore the idea of receptive fields.

Shared Weights

The main idea behind weight sharing (LeCun et al. (1998a)), is that each filter that is used on an input must extract the same feature from the entire input, and this is achieved by constraining the filters to have weights that do not change as they are used across an input. Sharing weights also reduces the number of parameters that have to be learned by the model.

Spatial or temporal subsampling

As stated in (Jarrett et al. (2009); Scherer et al. (2010)), the main motivation behind subsampling in CNNs is to improve robustness to small distortions and noise by reducing the resolution of the inputs.

4.2.1.1 Convolutional layers

Convolutional layers are the main building blocks of CNNs. To give a modified version of the input (which is commonly known as a feature map), the input to the convolution layer is convolved with a trainable filter, which is represented by an array of weights. In contrast to the NNs where inputs are represented as columns of neurons, inputs in CNNs are represented as rectangular blocks of neurons.

In order to ensure that the trainable filter used for the convolution mimics the small region of cells in the visual cortex that responds to certain regions of the visual field, it is constructed to

Table 4.1: Glossary of Key Convolutional Neural Network Terms

Key Term (Concept)	Explanation
Feature	Characteristics or traits in the input data. For example, if an input is a picture of a zebra, the features recognised by the network would be the zebra's stripes, two ears, and four legs etc.
Filter	Filters (also known as kernel or feature detector or neuron) act as feature detectors from the original input (i.e. the zebra image mentioned above). Filters are presented as a set of learnable weights which are learned using the backpropagation algorithm.
Feature maps	The feature maps of a CNN capture the result of applying the filters to an input image, i.e at each layer, the feature map is the output of that layer (i.e. modified input). All units in a feature map share the same set of weights and the same bias so they detect the same feature at all possible locations on the input.
Convolution	The mathematical combination of two functions to produce a third function. It merges two sets of information. The convolution is performed on the input data with the use of a filter to then produce a feature map.
Receptive field	The region of the input space that affects a particular unit of the network. Note that this input region can be not only the input of the network but also output from other units in the network, therefore this receptive field can be calculated relative to the input that we consider and also relative the unit that we are taking into consideration as the receiver of this input region. Usually, when the receptive field term is mentioned, it is taking into consideration the final output unit of the network (i.e. a single unit on a binary classification task) in relation to the network input (i.e. input image of the network).
Subsampling	The function of subsampling is to continuously reduce the dimensionality of each feature map in order to reduce the number of parameters and computation and the sensitivity of the output to shifts and distortions in the network, but retains the most important information. This shortens the training time and controls over-fitting. Subsampling can be of different types, i.e. Max, Average, Sum etc.
Weight Sharing	The function of the weight sharing technique is reducing the number of free parameters, thereby reducing the gap between test errors and training error.

be smaller in dimension than the input. Due to its smaller size, the filter cannot cover the whole input dimension at once. The convolving then happens region by region whereby the filter is shifted systematically until the entire input is covered. The region covered by the filter at each instance of the convolution operation is known as receptive field (Dlamini, 2018).

Also, each filter uses the same array of weights to sieve through the input, this is to ensure that a filter responds to a specific feature of the input. Therefore, features of the input are detected using different filters, as a result the number of feature maps created from a given input is determined by the number of filters used. The filter dimension and number of filters are parameters that need to be determined. The illustration of the convolution operation is shown in Figure 4.1 (Dlamini, 2018).

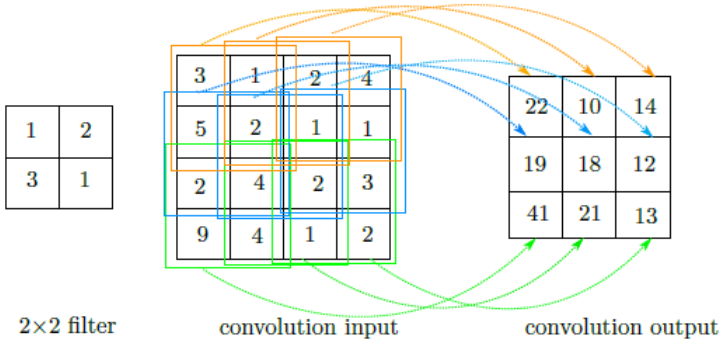


Figure 4.1: Illustration of the convolution operation. The filter is shifted throughout the input neurons and at each region of the input, the product is computed between the filter and input. The products are then summed together to get a single number. Focusing on the bottom right corner of the input, the convolution at that location gives: $1 \times 2 + 2 \times 3 + 3 \times 1 + 1 \times 2 = 13$

4.2.1.2 Subsampling layers

The dimensionality reduction of the feature maps obtained from previous layers is performed by pooling layers (Scherer et al. (2010)). Subsampling involves windowing each feature map into (either overlapping or non-overlapping) $n \times n$ regions, and returning a single value for each window. The size of the window is one of the parameters that must be set during the training phase of the CNN, and is arbitrary. Max-pooling and average pooling are two common subsampling operations discussed in literature.

If we let $x_i^{n \times n}$ be input i in region $n \times n$, and let x_j be the output of the subsampling operation, then the max-pooling function given by $x_j = \max_i x_i^{n \times n}$ computes the maximum value in each windowed neighbourhood whilst the average pooling function $x_j = \frac{\sum_i (x_i^{n \times n})}{n \times n}$ computes the average value (Scherer et al. (2010)). A common generalisation of the average-pooling is given by the sub-sampling function $x_j = f(\beta \sum_i x_i^{n \times n} + w)$ with a trainable weight β, trainable bias b and non-linearity $f(\cdot)$ (Scherer et al. (2010)).

4.2.1.3 Dimensions of the Convolutional NN

CNNs are described as being n-dimensional (nD), with n being the number of directions the convolution operation is calculated. Therefore, different types of CNNs exist, and the type to use depends on the dimensionality of the input and the objective of fitting the model. A 1D convolution may be applied to a 2D input where the convolution calculation happens in only a single axis and spans the entirety of the other axis. A 1D convolution may also be applied to a 1D input, similarly, a 2D convolution may be applied to 2D input or 3D input, where it spans the entire depth of the input volume in the latter case (Dlamini, 2018).

The forward and backward pass equations given in this chapter are those of a 2D-CNN from which equations for the 1D type can be derived by restricting convolution to one dimension instead of two.

Forward pass

A convolutional layer of a 2D CNN is given by:

$$\begin{aligned} s_{(i,j)}^{(n,l)} &= \sum_{h,k,m} w_{(k,m)}^{(h,n,l)} x_{(i+k,j+m)}^{(h,l-1)} + w_{(0)}^{(n,l)} \\ x_{(i,j)}^{(n,l)} &= f(s_{(i,j)}^{(n,l)}), \end{aligned} \tag{4.1}$$

where

a) $x_{(i,j)}^{(n,l)}$ denotes the output from the unit at position (i, j) of the n^{th} feature map in the l^{th} layer

b) $w_{(k,m)}^{(h,n,l)}$ denotes the weight at position (k, m) which connects the h^{th} feature map on the $(l-1)^{th}$ layer to the n^{th} feature map on the l^{th} layer

c) $w_{(0)}^{(n,l)}$ is the bias unit on the n^{th} feature map on the l^{th} layer

d) $s_{(i,j)}^{(n,l)}$ is the sum of the activation of the unit at position (i, j) of the n^{th} feature map in the l^{th} layer

e) $f(\cdot)$ denotes the activation function

For a pooling layer, if the convolution layer has m_1 feature maps, then the output of the pooling layer which normally follows the convolutional layer will also consist of m_1 feature maps but of smaller resolution. On the other hand, for a fully-connected layer, if a fully-connected layer follows a pooling layer with m_1 feature maps, then there would be weighted connections from every neuron or unit of the pooled feature maps to each neuron of the fully-connected layer.

Backward pass

CNNs are also trained using backpropagation (just like we did for ANN in Chapter 1), but with minor adjustments to incorporate weight sharing. For convolutional, fully-connected and

sub-sampling layers we follow the backpropagation algorithm. However, the max-pooling layer is treated differently in that the error is only propagated onto the unit where the maximum occurred during the forward pass (Scherer et al. (2010)).

The derivative of the objective function with respect to the weights can be expressed as follows:

$$\begin{aligned} \frac{\partial E}{\partial w_{(k,m)}^{(h,n,l)}} &= \sum_{i,j} \frac{\partial E}{\partial s_{(i,j)}^{(n,l)}} \times \frac{\partial s_{(i,j)}^{(n,l)}}{\partial w_{(k,m)}^{(h,n,l)}} \\ &= \sum_{i,j} \delta_{(i,j)}^{(n,l)} \times x_{(i+k,j+m)}^{(h,l-1)}, \end{aligned} \tag{4.2}$$

where $\delta_{(i,j)}^{(n,l)} = \frac{\partial E}{\partial s_{(i,j)}^{(n,l)}}$ are the network errors and the sum is over all neuron outputs $s_{(i,j)}^{(n,l)}$ where $w_{(k,m)}^{(h,n,l)}$ occurs. The weight sharing process can be summarized as:

$$\begin{aligned} \delta_{(i,j)}^{(n,l)} = \frac{\partial E}{\partial s_{(i,j)}^{(n,l)}} &= \frac{\partial E}{\partial x_{(i,j)}^{(n,l)}} \times \frac{\partial x_{(i,j)}^{(n,l)}}{\partial s_{(i,j)}^{(n,l)}} \\ &= f'(s_{(i,j)}^{(n,l)}) \times \frac{\partial E}{\partial x_{(i,j)}^{(n,l)}} \\ = f'(s_{(i,j)}^{(n,l)}) \times \sum_{n,q,u} \frac{\partial E}{\partial s_{(q,u)}^{(n,l+1)}} &\times w_{(q,u)}^{(h,n,l+1)}. \end{aligned} \tag{4.3}$$

4.2.2 Recurrent Neural Networks

Recurrent Neural Networks (commonly refereed to as RNNs) are an extension and improvement over Neural Networks (NN), they are a powerful type of NNs designed to handle sequence dependence, and are among the most widely used techniques in the statistical Natural Language Processing (NLP) work (Husken and Stagge, 2003). The recurrence in RNNs is defined by Kriesel (2005b) as "the process of a neuron influencing itself through any connection". One of the implicit assumptions made by the standard feedforward NN model is that the inputs (and outputs) are independent of each other. For example, in text classification, the input sentence "go to school" is totally equivalent to "school the go" according to NN, whereas these differ for CNN. This is considered the NNs major short coming as this assumption does not always hold true, especially when dealing with sequential data.

In cases when the assumption of independent inputs is violated, the recurrent neural network (RNN) should be used instead. The important feature that differentiates a RNN from standard NNs is the presence of at least one feedback connection that creates a loop or directed cycle between the nodes. It is this loop structure that makes it possible for information to persist in RNNs and therefore creating memory, making it suitable for sequence learning. A network may be able to compute sequential patterns that are complex, depending on the type of RNN or as Elman (1990) stated "be able to remember state through recurrent connections". Making

sequential information is the main idea behind RNN, this is done by making efficient use of temporal information in the output as sequence, for classification, and also for prediction. The recurrence in an RNN can happen in two ways (see Figure 4.2), *a*) self-connections from a node to itself across time, and *b*) connections between nodes in the hidden layer across time (Lipton et al., 2015). Both types of recurrences are assumed in this chapter.

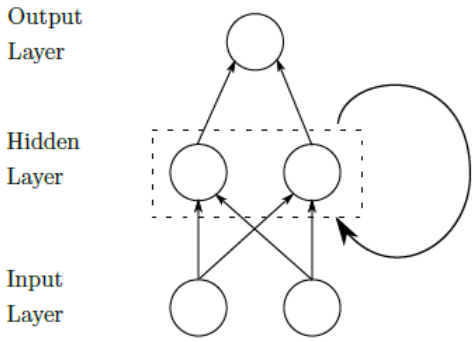


Figure 4.2: A RNN with two neurons in the input layer, two neurons in the selfconnected hidden layer and one neuron in the output layer. The loop represents both self-connections and connections between nodes across time.

Output neurons' activations can be used to classify a given input sequence, that is after the weights have been trained suitably. The arrangement of network (topology) that we use, is an extended Elman-Network (Stagge and Sendho, 1997), and it constitutes feed-forward, and a memory part. The memory part is responsible for storing the feedforward neurons from the previous time step, and also plays a role of additional input for the feedforward part. A time series classification starts with a warm up sequence, which sets the internal states correctly. After the warm up sequence, the subsequent time series is fed into the network, which results in the observation of the outputs. The "winner takes all" rule apply in every time step, that is, in every time step, the neuron with the highest activation provides the classification result. The rates of wins (i.e. rate at which the neuron had the highest activation over others) also provide information about the reliability of the classification; if the values of the rates are similar, the classification is not reliable, but if one dominates over all others, then the RNN is quite decisive (Husken and Stagge, 2003).

At a given time t, nodes with recurrent connections receive a signal, not only from the external input nodes, but also from the hidden node values in the network's previous state (Lipton et al. (2015)). If a RNN can be unrolled through time, then what would result is an architecture that is similar to a deep feedforward NN with the only difference being that the weights of the RNN are shared through all the time steps.

Forward pass

The forward pass of the network is defined by the following equations, where the second summation in Equation 4.4 is non-zero for hidden layers with recurrence:

$$s_{(j)}^{(l,t)} = \sum_{i=0}^{d^{(l-1)}} w_{(i,j)}^{(l)} x_{(i)}^{(l-1,t)} + \sum_{h=0}^{d^{(l)}} w_{(h,j)}^{(l,t-1)} x_{(h)}^{(l,t-1)}$$
$$x_{(j)}^{(l,t)} = f(s_{(j)}^{(l,t)}), \tag{4.4}$$

where

a) $d^{(r)}$ represents the number of nodes in the r^{th} layer

b) $x_{(i)}^{(q,t)}$ is the value of the input in the i^{th} neuron of the q^{th} layer at time t

c) $s_{(j)}^{(q,t)}$ the input to unit j of the q^{th} layer at time t

d) $w_{(i,j)}^{(l)}$ weights of connections from the i^{th} unit of the $(l-1)^{th}$ layer to the j^{th} unit of the l^{th} layer.

The information signal feeding into the output units can be computed using the following equation:

$$s_{(j)}^{(L,t)} = \sum_{i=0}^{d^{(l)}} w_{(i,j)}^{(l)} x_{(i,j)}^{(l,t)}. \tag{4.5}$$

Backward pass

There are two algorithms that have been developed to compute gradients of the objective function in a RNN setting. The first was introduced by Robinson and Fallside (1987), and is called real time recurrent learning (RTRL). The second algorithm, known as backpropagation through time (BPTT), was introduced by Werbos (1990), and it can be seen as generalisation of the backpropagation that is used to train FFNN. The latter is mostly used to train RNNs, because it is computationally more efficient and conceptually easier (Werbos (1990)). In this chapter, we thus focus on BPTT algorithm.

The BPTT propagates the model error starting from the output layer into the hidden layers (just like in backpropagation for FFNN). However, the major difference is that in BPTT, the output is not only affected by the outputs of the hidden layer at the current time step, but also outputs of the hidden layer from all previous time steps. In order to compute the contribution of the hidden layer output to the model error, the errors therefore have to be propagated through all the past times steps, hence the name backpropagation through time.

To train an RNN through BPTT is like unrolling the network into a larger feedforward network with duplicated weight parameters, where the size of the FFNN is dependent on the length of the sequence being learned. The longer the sequences being learned, the deeper the unrolled

feedforward network becomes and the more computations needed to backpropagate the errors in time. Following the BBPT process, the equation that before was $\delta_i^{(l,t)} = \frac{\partial E}{\partial s_i^{(l,t)}}$ is now:

$$\delta_i^{(l,t)} = f'(s_i^{(l,t)})(\sum_{j=0}^{d^{(l-1)}} \delta_j^{(l-1,t)} w_{ij}^{(l-1)}) + \sum_{h=0}^{d^{(l)}} \delta_h^{(l,t+1)} w_{ih}^{(l)}. \tag{4.6}$$

and defines the error on the i^{th} node of the l^{th} layer at time t.

The full sequence of δs can be computed starting at $t = T$, and recursively applying Equation 4.6 backward in time. Since the weights are shared by the network at all time steps,

$$\frac{\partial E}{\partial w_{ij}^{(l)}} = \sum_{t=0}^{T} \delta_j^{(l,t)} x_i^{(l,t)}. \tag{4.7}$$

There are different types of RNNs, i.e. vanilla RNN, which is just a regular RNN trained using a Vanilla Backpropagation; Bidirectional RNNs, which are based on the idea that the output at time t may not only depend on the previous elements, but also future ones; Deep (Bidirectional) RNNs, which are similar to Bidirectional RNNs, the only difference being that now there are multiple layers per time step; and Long-Short-Term-Memory (LSTM) RNNs, with the main feature being the capability of learning long-term dependencies (Britz, 2015). For the purpose of this chapter we will look at the Gated Recurrent Unit (GRU), and its subtypes, Stacked Gated Recurrent Unit (SGRU) and Bidirectional Gated Recurrent Unit (BGRU) methods, and therefore going forward we shall concentrate on and discuss only these methods further.

4.2.2.1 Gated Recurrent Unit (GRU)

In theory, RNNs can infer relationships between inputs that are separated by an arbitrary number of time steps. This however has been found not to be the case in practice, as RNNs are instead limited to looking only a few steps back. The learning of the relationship between inputs becomes more difficult as the duration of dependencies increase between inputs. When RNNs are trained using gradient-based learning (i.e. algorithms such as BPTT), they fail to utilize information in long sequences. This stems from the use of some activation functions (that squash their input), which cause the gradients to either vanish or explode. The sigmoid function for example maps its input to the range [0,1], so its gradients are small, and almost zero at the tails. This problem is referred to as the long-term dependence problem, and it worsens with the increase in layers of a neural network (Bengio et al., 1994).

With RNN learning that is gradient-based, model receives weights that are proportional to the gradient of the objective function of the weights in each iteration of the algorithm. Since the updates are proportional to the gradients, if the gradients are small, so are the weight updates, leading to slow model learning. During back-propagation, gradients are computed through the chain rule for differentiation. This means that these small gradients are multiplied numerous

times, and decrease exponentially as the number of layers increase. As a result, front layers in a network will train the slowest, as they have the smallest gradients.

The solution to the vanishing and exploding gradient problems is to either train the RNN without using gradient information or varying the architecture of the standard RNN (Bengio et al., 1994). GRU is one of the most used RNN variants that were introduced (for the latter solution) to attend to the vanishing gradients problem. GRU was proposed by Cho et al. (2014) to make "each recurrent unit to adaptively capture dependencies of different time scales, using gating units that modulate the flow of information inside the unit" (Chung et al. (2014).

Effective tracking of long-term dependencies is the main idea behind GRU, this is done while the vanishing/exploding problems are being mitigated. GRU does this using reset and update gates (both which contain weight parameters to be estimated); the function of the reset gate (which sits between the previous activation and the next candidate activation) is to forget the previous state, while the update gate's function is to make a decision on how much of the candidate activation to use in updating the cell state. Rather than replacing the entire activation (as traditional RNN), GRU are able to keep memory from previous activations. This allows GRU, for a long time to remember features, and allow backpropagation to happen through multiple bounded nonlinearities. This reduces the vanishing gradients likelihood. GRU also allows the entire cell state to be exposed to other units in network, and carries out both these operations using its rest gate (Chung et al. (2014).

The activation of "h_t^j of the GRU at time t is a linear interpolation between the previous activation h_{t-1}^j and the candidate activation $\bar{h}_t^j$" (Schmittlein and Peterson, 1994) and is given by:

$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j \bar{h}_t^j, \tag{4.8}$$

where z_t^j is an update gate that decides how much the unit updates its activation (or content), and is computed by:

$$z_t^j = \sigma(W_z x_t + U_z h_{t-1}). \tag{4.9}$$

This is a linear sum between the existing and newly computed state, of which GRU has no mechanism to control the degree to which it is exposed, but exposes the whole state each time. The candidate activation is given by:

$$\bar{h}_t^j = tanh(W x_t + U(r_t \odot h_{t-1}))^j, \tag{4.10}$$

where r_t is a set of reset gates, $\odot$ an element-wise multiplication. When off (r_t^j close to 0), the reset gate effectively makes the unit act as if its reading the first symbol of an input sequence,

allowing it to forget the previously computed state (Chung et al., 2014). The reset gate (similarly to the update gate) is given by:

$$r_t^j = \sigma(W_r x_t + U_r h_{t-1})^j. \tag{4.11}$$

The most important feature of the GRU, which traditional RNN lacks, is the additive component of its update from t to $t+1$. The content of a unit or activation is always replaced by the traditional RNN with a new value from the current input, and previous hidden states. On the other hand, the existing content is kept by GRU, and on top of it new content is added. Thus, remembering a specific feature existence in the stream of the input for a long series steps becomes easy for each unit. Also, any important feature (decided by the update gate) will be maintained as it is (i.e. will not be overwritten) (Chung et al., 2014). As means of reducing the difficulty that is caused by vanishing gradients, shortcut paths are also created by this addition feature. These shortcut paths bypass multiple temporal steps (as such avoid passing through multiple, bounded nonlinearities), and thus allow easy backpropagation of the error without quick vanishing of gradients (Hochreiter, 1991). The graphical representation of GRU network is shown in Figure 4.3 (Kostadinov, 2017).

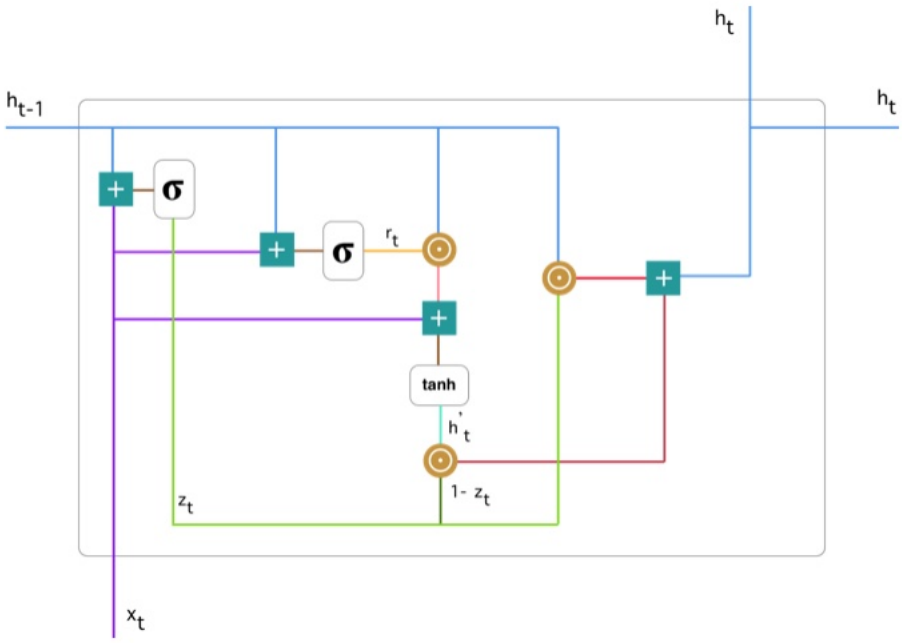


Figure 4.3: Graphical representation of GRU network; where z_t is the update gate, r_t the reset gate, $h_t^{'}$ the candidate activation (current memory content), and h_t is the final memory at current step

Stacked Gated Recurrent Unit (SGRU)

In order to increase the performance of the recurrent neural network model, one of the things to

be considered is increasing the capacity of the network. This is done by increasing the number of units in the layers or adding more layers. This is a classic way of building networks that are more powerful. In our case, the Stacked Gated Recurrent Unit (SGRU) is a GRU network whose capacity has been increased by both increasing the number of units in the layer, and also adding more layers (Chollet and Allaire, 2018).

Bidirectional Gated Recurrent Unit (BGRU)

Bidirectional RNN is one of RNN variants that can perform better than the standard RNN sometimes, depending on the task. RNNs are order dependent (unidirectional), meaning that they process time-steps of their input in order, and shuffling or reversing the time-steps can completely change the representations the RNN extracts from the sequence. The bidirectional RNN is a type of RNN that exploits the order sensitivities of RNNs. It consists of two regular layers, of which each processes the input in one direction (older time-steps first or newer time-steps first), and the representations of each layer are then merged. Since biderectional RNNs process sequences in both ways, they have an ability of catching patterns that may be overlooked by unidirectional RNNs (Chollet and Allaire, 2018). It is important noting that this only makes sense for some type of problems, where you see the whole sequence before classification. If you want to predict someone as yes/no only based on the sequence of inter-purchase times seen so far, then bidirectional RNNs are not suitable. In our case though the prediction is made only after the whole required sequence has been seen.

4.2.3 Model Accuracy

The idea behind building a model is that the model learns patterns in the data that generalize well for unseen data instead of just memorizing the data that it was shown during the training. So, once a model is built, it is important to check that the model performs well on unseen data that was not part of the training phase. To do this, the model is used to predict outcomes of the unseen (test) data, and compare the predicted responses to the actual outcomes (Mishra, 2018).

A number of metrics can be used to measure the predictive accuracy of the model, and the choice of accuracy metrics depends on the modeling task. The metrics we will be using in this chapter are misclassification, specificity, sensitivity, AUC (and Gini-statistic), F1-score and Kappa. As a hypothetical example, suppose that our test set has 1000 customers, and that a confusion matrix based on predictions of the test set is given by Table 4.2, with the following important terms: (a) True positives; cases which are predicted as 1 and are actually 1, (b) True negatives; cases which are predicted as 0 and are actually 0, (c) False positives; cases which are predicted as 1 and are actually 0, and (c) False negatives; cases which are predicted as 0 and are actually 1. The accuracy metrics can then be defined as following (we also work out each of the metrics using our example confusion matrix) (Mishra, 2018):

Table 4.2: Hypothetical Confusion Matrix for out test dataset, to be used as means of explaining the model accuracy measures

n = 1000	Predicted: 0	Predicted: 1
Actual: 0	962	14
Actual: 1	6	18

Misclassification Ratio

Misclassification is the ratio of incorrect predictions to the total number of input samples (see Equation 4.12). If for example our test set have 98% of responses as 0, and 2% as 1, then the model can get 2% misclassification rate by simply predicting all responses as 0s. Misclassification is a useful metric and one of the widely used accuracy measures, but can give a false sense of achieving high accuracy. The misclassification rate of 2% in our example might seem low, but when the cost of misclassifying minor samples is high, this becomes a huge issue. If for example we are dealing with a rare but fatal disease, the cost of failing to diagnose a sick person is much higher than a cost of wrongly classifying a healthy person as sick.

$$\begin{aligned} Misclassification\ rate &= \frac{False\ Positives + False\ Negatives}{Total\ number\ of\ predictions\ made} \\ &= (6+14)/1000 = 0.02 = 2\%. \end{aligned} \tag{4.12}$$

Sensitivity and Specificity

Sensitivity is the same as true positive rate (TPR), and Specificity is the same as false positive rate (FPR), and are given by:

$$\begin{aligned} Sensitivity &= \frac{True\ Positives}{False\ Negatives + True\ Positives} \\ &= (18)/(6+18) = 0.75 = 75\% \\ Specificity &= \frac{False\ Positives}{False\ Positives + True\ Negatives} \\ &= (14)/(4+962) = 0.014 = 1.4\%. \end{aligned} \tag{4.13}$$

F1 - score

In order to calculate the F1-score, we first need to define Precision and Recall. Precision is the proportion of the model's positive predictions that turned out to be correct, and Recall is proportion of all observed/true positives that were correctly predicted as such. The representations of Precision and Recall are:

$$\begin{aligned} Precision &= \frac{True\ Positives}{True\ Positives + False\ Positives} \\ &= (18)/(6+18) = 0.75 = 75\% \\ Recall &= \frac{True\ Positives}{True\ Positives + False\ Negatives} \\ &= (18)/(18+14) = 0.56 = 56\%. \end{aligned} \tag{4.14}$$

F1-score is then the balance between Precision and Recall and is mathematically given by Equation 4.15. The F1-score ranges between [0,1], and it tells us how precise and robust the model is, with higher the score, the better the model performance.

$$\begin{aligned} F1 &= 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}} \\ &= 2 * (1/((1/0.56) + (1/0.75))) = 0.64 = 64\%. \end{aligned} \tag{4.15}$$

Kappa

The Kappa statistic can be defined as the a measure of agreement between the observed and predicted outcomes. For example, our actual observations and predicted outcomes agree that 926 observations are positives, and 18 are negatives. If these ratings are randomly assigned, it is possible that some of these agreements could be by "chance", the numerical rating of the degree to which this occurs is given by a Kappa statistic. In order to calculate the Kappa-statistic, we first need to calculate Observed Agreement (A_o) and Expected Agreement (A_e), and these are given by:

$$A_o = True\ Positives + True\ Negatives. \tag{4.16}$$

$$A_e = \frac{(Actual\ Positives * Predicted\ Positives) + (Actual\ Negatives * Predicted\ Negatives)}{(Total\ Observations)}. \tag{4.17}$$

The Kappa-statistic is then given as:

$$\begin{aligned} Kappa &= \frac{A_o - A_e}{Total\ Observations - A_e} \\ &= (980 - 946)/(1000 - (946)) = 0.632 = 63\%. \end{aligned} \tag{4.18}$$

Area Under The Curve (AUC)

False Positive Rate (FPR) and True Positive Rate (TPR) both have values in the range [0,1]. Most models do not output discrete classification (yes or no); they produce probabilities of yes or no. All the FPR/TPR type metrics are based on discrete classification (yes/no), so we need some some way of converting probabilities into classes. Thresholds are the obvious way to do

this, but we can use any threshold we like (some will be better that others, and it is usually unknown which is best). The Receiver Operating Characteristic (ROC) curve is the name of the graph that results from plotting FPR vs TPR for all possible threshold values in the interval from 0 to 1, and the area under this curve is what is referred to as AUC. AUC is equal to the probability that the model will rank a randomly chosen positive example higher than a randomly chosen negative example. The actual value of AUC ranges between [0.5 and 1], and a Gini-statistic given by $Gini = (2 * AUC) - 1$ converts AUC to a value in the range [0,1], and is mostly preferred over AUC due to its easer interpretation in terms of proportions (i.e. 0 to 100% instead of 50 to 100%).

4.3 Data and Implementation

4.3.1 Data Description

The full dataset that we looked at in this chapter consists of over 300 000 customers, 144 unique stores, and over 3000 different products and a purchase history that contains transactions for all customers for a period of at least 1 year prior to the customer being offered incentives. We then extracted two subsets of the full dataset, one consisting of just over 1500 customers that have previously purchased the brand before, and the other consisting of just over 4000 customers whom have previously purchased in the department to which the product they received an offer for belongs. Our dataset consist of a series of inter-purchase times at this department for each customer, with a binary indicator of whether they became a repeat customer (1) or not (0). The minimum inter-purchase times length for brand and department customers were 9 and 10 with maximum lengths being 415 and 416, respectively. The average inter-purchase times were 16 and 4, with the up-sell (product customers that became repeat customers) and cross-sell (department customers that became repeat customers) ratios being 37% for both brand-level and department-level customers, respectively.

4.3.2 Methodology

Using the sequence of inter-purchase times as explanatory variables, the models try to predict which customers should be classified as 1 (repeat customers) or 0 (non-repeat customers). We split our dataset into training and test datasets through random sampling that ensures that the distribution of the response field is similar between the datasets. The training datasets will consist of 80% of the customers, and the rest will form the test dataset. We will then use the training dataset to fit the models, and use the fitted models to predict the outcomes of the test dataset. The predicted outcomes will then be compared to the actual outcomes to measure predictive accuracy. The predictive accuracy will be measured using the accuracy or performance measures mentioned above (i.e. misclassification, specificity, sensitivity, Gini-statistic, Area under the curve (AUC), F1-score and Kappa). The models were fitted in R,

using the *keras* package.

4.3.3 Model Implementation

4.3.3.1 Model Architectures

In total, 4 deep learning models were used to predict repeat customer, and these models had the following properties:

- CNN model - with one convolution layer, one max-pooling layer, one dropout layer, and one hidden layer. All filters were 1-dimensional.
- GRU model - with one GRU layer, and 32 units
- SGRU model - with two GRU layers, and 32 units
- BGRU model - with one bidirectional GRU layer, and 32 units

4.3.3.2 Preprocessing and Parameter Search

Padding

Our input data is a sequence of inter-purchase times, whose length is equal to one less than the number of department visits a customer made, and whose entries are the number of days between consecutive department visits. For each customer the length of the sequence will differ, resulting to a dataset that has sequences of different lengths. The models fitted in this chapter work better if the data contains sequences of the same length, as such we padded our data such that all sequences are of the same length. This involved finding the sequence with the longest length, and making all other sequences have the same length by putting a series of 0s equaling this maximum length minus sequence length of a particular customer in front of the first sequence entry.

Parameter Search

Standard neural networks in general are hard to configure, because they require setting many hyperparameters. For an FFNN for example, hyperparameters that need setting include number of hidden layers, number of nodes in each hidden layer, learning rate and type of optimisation techniques (Dlamini (2018). As the NNs get more complex the number of parameters increase, as such for the methods considered in this chapter, more parameters need to be set. It is inefficient and computationally impossible to exhaust all possible combinations of hyperparameters, therefore only a few can be (and were) considered. For this reason, hyperparameters optimisation is an active area of research where more efficient methods of hyperparameters optimisation are being investigated and proposed, with some of the publications exploring this issue being Bergstra et al. (2011) and Bergstra and Bengio (2012).

In this chapter, our models were were trained for 30 epochs, using the *keras* package in R. Models were trained under numerous parameter configurations, and configurations with the highest performance accuracy were used. The performance accuracy results and evaluation of some hyperparameters combinations are presented in the following results section.

4.4 Results and Discussion

As explained in Section 2, the models are fitted on two different sets of data, one which consists of customers that have previously purchased the brand for which they received an offer, but looking at their departmental inter-purchase times. The second set of data consists of customers that have not necessarily purchased the brand before, but have purchased in the department to which the brand belongs. The results presented in this chapter will be for both types of customers, those that were targeted for up-sell purposes (brand customers) and those that were targeted for cross-sell purposes (department customers). The first set of results is mainly for the purpose of comparing the significance of the models; how the sequence models fare with the traditional models, and which model fitted our dataset the best. The second set of results is focusing on demonstrating the effect of changing some hyperparameters to the models. The third set of results will be focusing on practical interpretations of the results, and how such models can be applied in practice.

4.4.1 Comparing Classical Classifiers to Deep Learning Methods

The Tables 4.3 and 4.4 contain accuracy metrics (7 in total) that were explained in Section 2 for all the models that were fitted (6 in total) for brand and department customers, respectively. Figures 4.4 and 4.5 show ROC curves for all models, for brand and department customers, respectively. These results are mainly for comparing the performance of the sequence models (CNN, GRU, SGRU and BGRU) to that of the traditional models (Tree and ANN). We assess whether the sequence models are indeed better predictors when a sequence input is used. This would give an idea of whether it is worth it applying these models over the traditional ones, or whether using any model adds any value at all.

Table 4.3: Accuracy measure metrics for all models - brand customers

Metric	Tree	ANN	CNN	GRU	SGRU	BGRU
Missclassification	40.2	37.54	37.21	37.21	37.21	36.23
Gini	1.1	13.6	15.7	21.8	21.2	21.9
Sensitivity	10.7	55.4	85.7	75	75.9	64.3
Specificity	92.1	57.7	31.7	44.4	42.3	54
AUC	50.5	56.8	57.9	60.9	60.6	60.9
F1 Score	17.3	48.8	17	55.8	55.6	53.1
Kappa	3.3	12.4	14.7	17.1	15.9	16.8

Table 4.4: Accuracy measure metrics for all models - department customers

Metric	Tree	ANN	CNN	GRU	SGRU	BGRU
Missclassification	38.97	40.93	38.97	38.97	38.97	38.97
Gini	0.3	-0.1	9.3	3.1	3	2.6
Sensitivity	0.9	59.1	32.4	51.9	29.2	40.9
Specificity	99.4	44	77.1	53.2	78.7	66.3
AUC	50.2	50	54.7	51.5	51.5	51.3
F1 Score	1.9	39	38.5	46.1	36	42.2
Kappa	0.4	-3	10.1	4.9	8.5	7.2

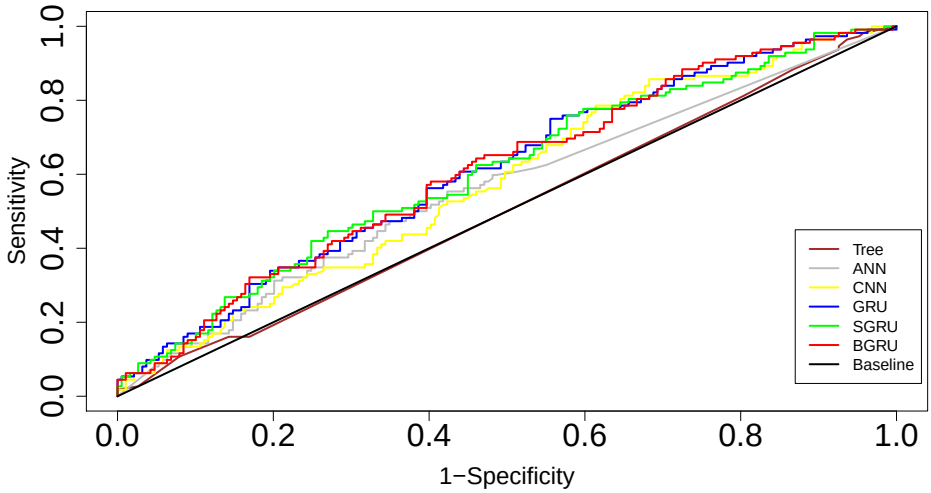


Figure 4.4: ROC curve for all 6 models (brand customers), the black line represents the baseline (i.e. AUC=50), the model that is furthest (with the biggest AUC) from this line is the best model for this dataset, with the model closest to this line the worst)

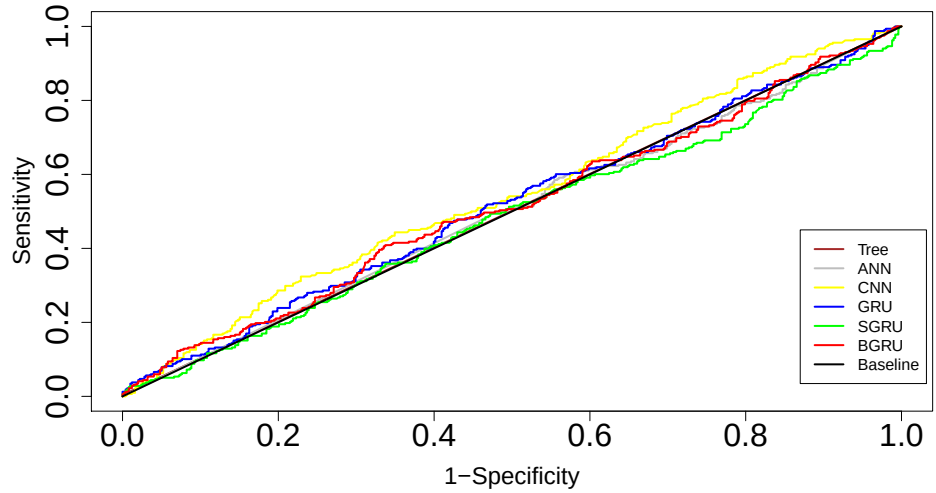


Figure 4.5: ROC curve for all 6 models (department customers), the black line represents the baseline (i.e. AUC=50), the model that is furthest (with the biggest AUC) from this line is the best model for this dataset, with the model closest to this line the worst)

The key findings from the results presented in Table 4.3 and Figure 4.4 are that, there is nothing much separating the main models, as such, depending on the metric used, conclusions reached can be completely different. Based on the study by Green and Milne (2010) where they discuss the advantage of using the Gini-statistic over other methods used to assess model performance by both researchers and practitioners for direct marketing, in this study we mainly use the Gini-statistic. The main observations from the model comparison results (see Tables 4.3 and 4.4, and Figures 4.4 and 4.5) are that, inter-purchase times only help to predict behaviour when they are about the same product . If inter-purchase times are from a different product in the same department all models do no better than a null model (see Table 4.4 and Figure 4.5). Secondly, for the same-product inter-purchase times, it is worth taking the sequential nature of data into consideration, as the deep learning methods that take sequence order into consideration (CNN, GRU, BGRU and SGRU) when training outperform classical classifiers that do not (Tree and ANN).

The models were fitted on a dataset that has 37% proportion of customers (from our subset) that continued purchasing the product after it expired. So based on this, a model that just predicts everyone to be non-repeaters would have a misclassification ratio of 37%. This means that we can expect our models to have misclassification rates around this ratio (37%), with better models having a lower misclassification rate, and poor ones having higher misclassification ratio. The brand customer results in Table 4.3 show that this is the case with most of our sequence models (i.e. ANN, CNN, GRU and SGRU), this is sightly higher for Tree, and slightly lower for BGRU. The specificity and sensitivity show that the Tree classification model's performance is very poor

(high specificity and low sensitivity), and different to that of sequence models (both specificity and sensitivity closer to 50% than to 100% and 0%, respectively). This means that the Tree put everything into the majority class, which is not very useful. The deep learning models are much more balanced. Interestingly, the ANN is better balanced compared to the Tree model, and more similar to the deep learning models.

The AUC and Gini results paint a clearer picture, that the deep learning models perform better than the traditional models. This is evident when we compare Tree results to that of BGRU in particular on the Gini-statistic, with BGRU being over 20% better than random, and Tree not any better. There is not that much difference between the GRU methods though, suggesting that stacking the GRU network or making it train from both directions may not be beneficial for this dataset. An interesting results though is how the ANN results are similar to that of CNN, and very different to that of Tree. Since ANN and Tree do not take the sequence of the data into consideration when training, one would expect these to perform similarly. However the results show otherwise, which could suggest that even though ANN was trained to perform the classifications in the traditional way, it managed to discover some hidden complicated patterns as neural networks are themselves part of the deep learning methods. The results in Figure 4.4 show the same outcome as well, that GRU, SGRU and BGRU have very similar results, with BGRU being slightly better (doing 22% better than random), with CNN doing better than the traditional models, but not as good as the GRUs. Trees on the other hand seem to not be doing better than random, suggesting that using Trees for such a classification would not yield any value.

The models however seem to perform poorly on the department customers data (see Table 4.4 and Figure 4.5, with the Gini-statistic ranging between 0 - 9%, with CNN having the highest. It is obvious from these results that the models find it easier to classify customers that have bought the product before than those who have not, and therefore not worth it discussing these department results further. In practical terms, this suggests that this type of modeling is only beneficial for up-selling purposes, and not cross-sell.

Overall on the brand customers dataset results, there is nothing much separating the main models as they are all performing similarly (Gini-statistic ranging between 16-22%). Though these results are not poor, they are not convincing either. These results show that, even though these methods are highly regarded when it comes to sequence modeling, the output that is being modeled is vital for their performance. They have been proven to perform well for speech translation and weather forecast for example, with the output in those cases being the same (format) as the input. Maybe using inter-purchase sequences to predict a binary outcome is not suitable for these models, maybe if we were predicting the next (say couple of) inter-purchase time(s) they would have performed better. With that said, the sequence models still outperform non-sequence models on sequence data, even though the results were not convincing.

These results confirm some common sense intuitions, that customers who have purchased the brand before, are more likely to become repeat customers than those who have not. Based on these results, we can conclude that scorecard development would be beneficial for these retailers, and utilizing these models could help improve campaign results by converting more customers, and thus generating more profit. Store managers could make use of these scorecards as part of strategy, or lead with them, track the performance overtime, and retrain the models if and when needed to.

4.4.2 Evaluation of Hyperparameters

In this section we analyze the effects of some of the hyperparameters on the Gini-statistic. The hyperparameters were tried and tested only for the main models (i.e. CNN and GRU models). The optimal configurations obtained for the basic GRU model were then also applied to the SGRU and BGRU models. The four hyperparameters analyzed in this chapter are: (a) activation function, (b) optimization function, (c) batch size and, (d) learning rate. The results for CNN hyperparameter search are shown in Table 4.5, and the results for GRU are in Table 4.6

Effect of Activation Function

Relu, sigmoid, tanh and softmax are the activation functions evaluated in this chapter. The results produced by these activation functions for the CNN models were very similar, with the average Gini-statistic ranging between 3.5% - 5.2%, with tanh having the most, while sigmoid resulted to the least. The Gini-statistic values are quite high for GRU, so the differences are bigger, but the biggest 2 are similar 18.3% and 15.7% for sigmoid and tanh, respectively. Relu produced an average Gini-statistic of 7.8%. On the other hand softmax seemed not to be compatible with training GRUs as it resulted to Gini-statistic of zero for all configurations considered.

Effect Optimisation Function

The three optimization methods evaluated in this chapter are adam, gradient descent (sgd) and RMSPRop. The results show that the optimization methods did not produce completely different results, but sgd slightly outperformed the other two methods for CNN with the average Gini-statistic of 5.7% compared to 3.8% and 3.6% of adam and RMSPRop, respectively. Similarly for the GRU results, the effect of optimazation methods did not differ considerably, but adam resulted to the highest Gini-statistic average (12%), while RMSPRop and sgd are 10.5% and 10 8.9%, respectively.

Effect Batch Size

The batch sizes considered in this chapter were 20, 50 and 100. Batch size did not seem to have

major impact on the Gini-statistic as the average Gini-statistic was similar for all of them, i.e. 4.4%, 3.8% and 5.0% for 20, 50 and 200 batch sizes respectively. These same results for GRU were 12.8%, 9.5% and 9.1% respectively, making batch size 20 the more optimal for GRU.

Effect Learning Rate

This chapter involved evaluating these three learning rates, 0.01, 0.001 and 0.0001. These resulted to very similar average Gini-statistic for GRU with the difference being 1% between the highest and lowest. These were even more similar for CNN, with the same difference being less than 1%.

So, after all the hyperparameters were evaluated and analysed, the configuration that resulted to the highest Gini-statistic was used for both our models. These optimal combinations of the hyperparameters are highlighted (shaded) in the respective table results for CNN and GRU. This configuration for CNN was *softmax* activation function, *sgd* optimization method, batch size of 20 and learning rate of 0.0001. This combination for GRU was *sigmoid* activation function, *adam* optimization method, batch size of 20 and learning rate of 0.01.

Table 4.5: Validation Gini-statistic values achieved by CNN

	optimiser = adam								
	Batch = 20			Batch = 50			Batch = 100		
Activation function	lr = 0.01	lr = 0.001	lr = 0.0001	lr = 0.01	lr = 0.001	lr = 0.0001	lr = 0.01	lr = 0.001	lr = 0.0001
relu	0.1%	7.2%	3.4%	8.2%	4.6%	3.2%	4.8%	2.6%	6.4%
sigmoid	0%	-1.2%	1.2%	2.3%	3.3%	4.1%	3%	3.2%	6.1%
tanh	7.5%	9.1%	9.7%	5.9%	1.3%	2.7%	3.1%	3.5%	5.5%
softmax	1.8%	2.1%	2.1%	2.5%	2.9%	2.8%	4.2%	2.7%	5.8%
	optimiser = sgd								
	Batch = 20			Batch = 50			Batch = 100		
Activation function	lr = 0.01	lr = 0.001	lr = 0.0001	lr = 0.01	lr = 0.001	lr = 0.0001	lr = 0.01	lr = 0.001	lr = 0.0001
relu	5.8%	6.4%	3.1%	10.2%	4.4%	2.6%	12%	-5.3%	6.3%
sigmoid	5.2%	4.8%	4.1%	5.9%	10.3%	3.6%	6.7%	3.7%	7.3%
tanh	4.8%	5.3%	5.4%	5%	5.7%	6.2%	5.6%	6.6%	3.8%
softmax	1.9%	6.2%	**16.9%**	2.6%	7.1%	4.8%	8%	12.4%	1%
	optimiser = rmsprop								
	Batch = 20			Batch = 50			Batch = 100		
Activation function	lr = 0.01	lr = 0.001	lr = 0.0001	lr = 0.01	lr = 0.001	lr = 0.0001	lr = 0.01	lr = 0.001	lr = 0.0001
relu	5%	7.2%	5.7%	6.6%	-5.5%	-1.6%	5.1%	5.9%	4.8%
sigmoid	0%	0%	0%	1.9%	2%	3.2%	5.4%	0.5%	8.5%
tanh	11.7%	15.5%	-5.1%	8.4%	-2.4%	0.9%	4.5%	4.4%	5.9%
softmax	0%	1.9%	1.9%	2.7%	3.4%	3.8%	5.4%	6.3%	5.4%

4.4.3 Illustrating application of the models in practice

The purpose of the following set of results (shown in Tables 4.7,4.8 and 4.9) is to show how such models can be used in practice, with Table 4.7 focus being customer targeting, wheres the latter 2 focus on customer profiling. It is from these tables that practical interpretations of actionable insights can be generated for managers. In marketing, these models would be used by creating what is known as a "scorecard", which is an attempt to provide a quantitative estimate

Table 4.6: Validation Gini-statistic values achieved by GRU

	optimiser = adam								
	Batch = 20			Batch = 50			Batch = 100		
Activation function	lr = 0.01	lr = 0.001	lr = 0.0001	lr = 0.01	lr = 0.001	lr = 0.0001	lr = 0.01	lr = 0.001	lr = 0.0001
relu	21.7%	18.3%	14.9%	20.9%	18.3%	3.5%	-1%	14.9%	16.2%
sigmoid	26.8%	**22.6%**	21.1%	19%	18.5%	17.6%	18.1%	15.1%	15%
tanh	20.9%	15.3%	20.2%	19.1%	14%	4.7%	13.2%	4.1%	17.5%
softmax	0%	0%	0%	0%	0%	0%	0%	0%	0%
	optimiser = sgd								
	Batch = 20			Batch = 50			Batch = 100		
Activation function	lr = 0.01	lr = 0.001	lr = 0.0001	lr = 0.01	lr = 0.001	lr = 0.0001	lr = 0.01	lr = 0.001	lr = 0.0001
relu	0%	20.3%	0%	0%	0%	0%	0%	3.3%	0%
sigmoid	21.1%	12.2%	21.1%	10.9%	21.1%	21.4%	20.8%	0.1%	21.3%
tanh	19.8%	20.5%	21.1%	21.2%	0%	0%	21.1%	20.9%	20.5%
softmax	0%	0%	0%	0%	0%	0%	0%	0%	0%
	optimiser = rmsprop								
	Batch = 20			Batch = 50			Batch = 100		
Activation function	lr = 0.01	lr = 0.001	lr = 0.0001	lr = 0.01	lr = 0.001	lr = 0.0001	lr = 0.01	lr = 0.001	lr = 0.0001
relu	0%	21%	3.6%	0%	17.9%	0%	0%	15.9%	0%
sigmoid	21.4%	23.1%	22.2%	19.3%	17.9%	17.1%	17.4%	17.4%	14.7%
tanh	15.6%	17.4%	19.3%	21.1%	18.1%	20%	9.2%	13.5%	16.6%
softmax	0%	0%	0%	0%	0%	0%	0%	0%	0%

of the probability that a customer will display a defined behaviour (i.e. in this case, become a repeat customer). This involves producing scores for all eligible customers in the database. The traditional way of doing this is by assigning each customer a probability, and sort the database by the probability (ascending or descending depending on whether the task requires highly likely or unlikely customers), and target the top customers.

In most cases, scorecards are used as part of the strategy (i.e. in conjunction with other ideas and initiatives), but can also be used as the only plan. It would be costly for retailers to send offers to all customers, so a small subset is ideally selected. Without any predictive modeling, these customers are randomly selected, the main idea behind scorecard development is to improve this sampling strategy by selecting only customers with high likelihood of converting. One of the success measures of such predictive models is what is known as lift, which is the difference between the conversion rate of the model selection and that of the random selection. For example, the conversion rate of the random selection for our dataset is 37%, so the lift will be the model selection's conversion rate minus 37%, which needs to be positive for the models to be adding any value.

We established that department results are not good at all, and should not be used in practice as this would not add any value. We will thus not include these results for the practical illustration. Also, only the best model would be used in practice, so the results in this section will only be based on the the BGRU results, which is the best model for our dataset. The results in Tables 4.7, 4.8 and 4.9 are generated from a scorecard that is developed from probabilities. We assigned all customers a probability, then sorted customers in descending order for all models. This thus produce a ranking, whereby the customer with the highest score ranks first, and the customer

with lowest probability ranking last. Out test dataset has 301 customers, meaning that our rank ranges between 1 and 301, with 1 being the highest rank, and 301 the lowest. This set of results thus focuses on the ranking rather than the actual probability, as sometimes the actual probability is not important, but the order is, especially when it comes to targeting top ranking customers.

In Table 4.7, the customers are ranked by score, and broken down into 4 blocks of 25% of customers each, starting from the top 25% to the bottom 25%. That is, the top 25% of the customers will be those ranking between 1 and 75, the next 25% containing those with ranks between 76 and 150. The last two bands constitutes customers ranking between 151-225, and 126-301, respectively (i.e. all bands contain about 75 of respondents). The following tables are also based on these bands, whereby in Table 4.8 these are shown per average inter-purchase times bands, and per sequence length bands in Table 4.9.

Table 4.7: Proportion of customers ranking from top to bottom by score, showing the lift produced by the model per score-band and overall

Ranking	Repeat Indicator					Lift
	No	Yes	Total	%No	%Yes	
1 - 25%	38	37	75	51	49	12
26 - 50%	45	30	75	60	40	3
50 - 75%	49	26	75	65	35	-3
76 - 100%	57	19	76	75	25	-12
Grand Total	189	112	301	63	37	0

Table 4.8: Proportion of customers ranking from top to bottom by score per average inter-purchase times band, showing the average inter-purchase times per score-band

Inter-times	Counts				Proportions			
	1 - 25%	26 - 50%	50 - 75%	76 - 100%	1 - 25%	26 - 50%	50 - 75%	76 - 100%
<= 10	19	1	0	0	25%	1%	0%	0%
11-20	39	25	20	6	52%	33%	27%	8%
21-30	14	35	30	30	19%	47%	40%	39%
31-40	3	11	22	39	4%	15%	29%	51%
> 40	0	3	3	1	0%	4%	4%	1%

The results in Table 4.7 show that even though the models overall classify customers similar to random selection (based on misclassification rate), the models actually do a better job in classifying the top ranking customers. These results suggest that targeting top customers based on these models would yield some value. This is evident when we look at the lift column in particular, as the top 25% customers would yield 12% lift (i.e. 49% correct model classifications minus 37% correct random selections). The grand total row shows that, if all customers or a random subset were to be selected, a lift of 0 would be generated. Selecting the top customers brings about the most lift, including lower ranking customers decreases this till a lift of 0 when

Table 4.9: Proportion of customers ranking from top to bottom by score per sequence length band, showing the sequence length per score-band

Length	Counts				Proportions			
	1 - 25%	26 - 50%	50 - 75%	76 - 100%	1 - 25%	26 - 50%	50 - 75%	76 - 100%
<= 10	4	11	19	31	5%	15%	25%	41%
11-20	22	45	42	42	29%	60%	56%	55%
21-30	20	15	13	3	27%	20%	17%	4%
31-40	14	3	1	0	19%	4%	1%	0%
> 40	15	1	0	0	20%	1%	0%	0%

all customers are included.

Sometimes profiling customers can be useful for store managers. This is because despite knowing that it is high ranking customers who are likely to convert, store managers sometimes are interested in knowing who exactly are these people. The obvious way of answering this question is by profiling these customers on attributes that are easy to understand. These normally include demographical data (i.e. age, gender), geographical data (i.e. where they stay) and behavioral data (i.e. how they shop), and these attributes normally form part of the explanatory variables (i.e. input to the models). Our explanatory variable was inter-purchase times, so we profiled our customers on this, the sequence length and average inter-purchase times to be exact.

The results in Table 4.8 show that, customers who have shopped the department more frequently (i.e. shorter average inter-purchase times) have high likelihood of becoming repeat customers than those who shop less. This is evident when we compare the top and bottom 25% ranking customers, with the former having over 3-quarters of its respondents with average inter-purchase times less or equal to 20, this number for the latter being just 8%, with more than half of the customers in this band having average inter-purchase times greater than 30. The results in Table 4.9 show that customers with longer sequences are more likely to become repeat customers than those with shorter sequences. Comparing the same bands as above (top and bottom 25%) shows that almost all customers in the bottom quarter have sequence lengths shorter than 21, wheres the top quarter has only 35% of these customers. There are no customers that have visited the department more than 30 times in the bottom quarter, and this number is over a quarter for the bottom one.

These results confirm some common sense intuitions, that customers that are regulars in the department are likely to convert more than those who shop less, and that customers who have purchased the brand before, and are already loyal to the department, are highly likely to also be loyal to the brand they received an offer for. This set of results shows that profiling customers can also be useful in practice, as it is a form of segmentation that allows store managers to associate a certain type of people with a particular behaviour.

4.5 Summary

The objective of this chapter was to predict which customers are likely to become repeat customers after being given a special offer. This was done for both up-sell and cross-sell purposes. This involved building 6 models on department customer inter-purchase times, 4 deep learning models, and 2 traditional models. This chapter started with an introduction section, where we gave the background to the problem we would like to solve, explained exactly what is it that we were willing to achieve in this chapter, and concluded by outlining the chapter. The following section discussed literature review, starting with sequence modeling, and previous work on sequence modeling. Deep learning methods were then discussed, where we discussed Convolutional Neural Networks (CNN), and Gated Recurrent Units (GRU) and its subtypes in Stacked Gated Recurrent Units (SGRU) and Bidirectional Gated Recurrent Units (BGRU). This was followed by the discussion of the model accuracy and the metrics to measure it, and these metrics were misclassification, AUC, Gini-statistic, specificity, sensitivity, F1-score and Kappa. The dataset to be used was then discussed in the following section, together with the methodology that would be followed, and model implementation. In the previous section we presented and discussed our results, and concluded with recommendations.

The highlights from the results are that, for this type of modeling, inter-purchase times are only useful when they are about the same product, as models did no better than random if inter-purchase times were from a different product in the same department. This implied that, these types of models are only useful for up-sell purposes, and should not be used to promote cross-sell. Another observation was that it is useful to take the order of the sequence into account, as models that do this do better than those who do not, with the latter not doing any better than a null model.

The results showed that none of the models are convincing in terms of being able to distinguish between repeat and non-repeat customers, but the deep learning models (i.e. CNN, GRU, SGRU and BGRU) perform better than standard classification models (particularly decision trees). However, the results also showed that these models can be a useful tool for management when it comes to campaign targeting as the models showed they could produce some lift, especially by focusing on the top 25% ranking customers. The results also confirmed some common sense intuitions (i.e. customers with longer purchase histories, and those who shop often are likely to become repeat customers etc).

We also noted that the overall results were somehow poor, as the best model had only Gini-statistic of 22%. We did however argue that this could be due to the relationship between the input and output (i.e. sequence of inter-purchase and binary output). CNN also showed that if one is concerned about the computational time, they should be preferred over RNNs as they train much quicker. Stacking and adding the bidirectional element to the GRU caused the model

to train even much slower. The GRUs though did compensate for this long running time as they produced slightly better results than CNN (on the fundamental brand data). Concerning the results though, we believe that maybe modeling a different outcome (i.e. sequence of future inter-purchase times) could perhaps have yielded better model performance, as predicting a binary outcome (whether a customer will be a repeat customer or not) from inter-purchase times turned out to be complicated for the models.

Chapter 5

Conclusion

5.1 Summary and Recommendations

The aim of this dissertation was to mine a large shopping dataset for a retailer containing purchase histories of thousands of customers, for insights into the behaviour of those customers. This involved building 3 predictive models with each forming a chapter of the dissertation, namely; a model to forecast store future daily visits, a model to detect changes in consumer's purchasing behavior, and a model to predict repeat customers after receiving a special offer. Each model has its own chapter, and all the detailed explanations were discussed in the respective chapters, including data, literature review, implementation, results and recommendations. In this chapter we give an overview of each of the model building chapters, and as such summarizing the whole dissertation, after having given an introduction in Chapter 1.

The model built in Chapter 2 was for forecasting future daily visits a store should expect in the next 3 months. This was tackled by building in total 8 time-series forecasting models, namely; average models (simple, moving and weighted moving), exponential smoothing models (single, double and triple), ARIMA model and artificial neural networks. The primary purpose of this chapter was to compare the performance of the highly rated artificial neural networks with the more traditional ARIMA forecasting model, and even simpler forecasting heuristics such as the averaging and smoothing methods already mentioned. These models were fitted to a univariate time-series data, and were compared to each other using 3 accuracy measures, namely; mean absolute deviation (MAD), mean forecast error (MFE) and Diebold-Mariano significance test. The results showed that there was not much separating the methods, with ARIMA model and single exponential smoothing slightly performing better than neural networks, but neural networks outperforming the rest of the methods, with moving average being the worst method for our dataset. Based on these results, it would thus be recommended to the retailer that if building all the models would be costly and maybe time consuming, then only ARIMA and single exponential smoothing models should be applied, otherwise all models can be built and applied, with the best one being selected at each building time as there is not much difference between the methods.

In the third chapter, we built a model to detect changes in consumers' purchase behavior. This was tackled by using change-point models to detect changes in customers' inter-purchase times. This involved building 2 models in total, one fitted to store inter-purchase times (where we modeled time between store visits), and the other to brand-inter-purchase times (where we modeled time between product purchases). We discussed 2 types of change point models; single change-point model (which assumes that there is exactly one change point, and tries to estimate where it is), and multiple change-points model (which does not make this assumption, and instead also estimates the number of change points), and the latter form was implemented. Also discussed were various approaches that can be used to estimate the parameters of these models, and concluded that the Bayesian method is the widely used approach, and thus in the chapter we make use of this approach. The results in this chapter spanned over 3 sections, with the first section demonstrating the change point model results using simulated data, and the way the model responded here confirmed some common-sense intuitions. In the following section, we illustrated the use of the change-point model in practice, where we focused on practical interpretation and illustrated some of the patterns we found for some example sequences taken from the actual dataset. The results in this section were consistent with those obtained from the simulation data. In the following section we presented the main results, and the main observation was that, depending on the length of the sequences, minority to a handful of customers do experience changes in their purchasing behaviours, with the longer sequences having more changes than the shorter ones. This suggests that maybe store managers would need to set some cut-off value, for example say, a change-point will only be considered if it occurs within 30 purchases, and disregard anything happening after that as longer sequences are likely to have more change points than shorter ones.

In the fourth chapter, we built a model to predict which customers are likely to become repeat customers after being given a special offer. This was achieved by building a total of 6 models (2 classical and 4 deep learning) which were fitted to univariate time-series data. This time-series consisted of times between two successive purchases (i.e. inter-purchase times) in the department to which the product the customer received an offer for belongs. This input data is sequential in nature, so we used models that provide a good way for dealing with sequential inputs in convolutional neural networks (CNN), and recurrent neural networks (RNN). For CNN, we used the 1-dimensional (1D-CNN) subtype, whereas the gated recurrent unit (GRU) subtype was used for RNN, together with its subtypes in stacked gated recurrent unit (SGRU) and bidirectional gated recurrent unit (BGRU). These models were then compared to standard models that do not take the sequence of the data into account, in artificial feedforward neural networks and decision trees. Various accuracy measures were used to measure the accuracy of the models, and these were misclassification, specificity, sensitivity, area under the curve (AUC), Gini-statistic, F1-score and Kappa. The results showed that, inter-purchase times are only useful when they are about the same product, as models did no better than random if inter-purchase times were from a

different product in the same department. Secondly, it is useful to take the order of the sequence into account, as models that do this do better than those who do not, with the latter not doing any better than a null model. Lastly, while none of the models performed well, deep learning models perform better than standard classification models and produce some substantial lift.

Big data and data science has been growing fast in the industry in recent times, and all businesses soon will have to apply data science to generate more insights from the data. It has been shown that the data being generated currently grows at a more rapid rate compared to the data that was generated, say 20 years ago. As such many companies have a lot of data that they are not using for insights as they do not have the right systems and skills to be able to generate actionable insights from this massive data they already have available. Data science provides approaches to tackle difficult business questions through the use of complex mathematical and statistical methods (i.e. machine learning), and sophisticated systems and environments that are designed to manage big data (i.e. cloud computing). The goal of this dissertation was to examine some of the approaches in which data science can be used to tackle business problems that come about in the industry. We made use of some of the programming languages that are currently being widely used in R, Python and SQL, and also ran some of our scripts in cloud computing (i.e. AWS), so our research can be seen as a starting point to unlocking value from big data.

5.2 Limitations and Future Work

First of all, the data used in this research was very clean with minimal noise, and this is hardly the case in practice. The data was prepared and provided for a competition, and the transactions data we used was already in an easy-to-use format. In practice one normally has to spend more time in data clean-up than they in analysis, and models need to be as robust to that noisy data as they are to the clean data. In Chapter 2, we did not manage to include more machine learning methods to compare to the classical ones. In Makridakis et al. (2018) for example, multiple machine learning methods were used, and thus their findings were more robust than ours in terms of the comparison between machine learning and statistical models.

Our change-point models focus only on inter-purchase times, and do not consider other aspects of behaviour such as brand or store selection (i.e. similar to the study by (Clark and Durbach, 2014)). Considering these would have allowed us a chance to compare the performance of the models, and see whether there are any significant differences for our dataset. Also, only the Bayesian approach was followed, perhaps estimating the change-point parameters using other approaches could have been beneficial. In Chapter 4, our main aim was to apply sequential modeling and use only inter-purchase times as input to predict repeat customers, and this proved not to be very fruitful as the results were still not that convincing. Maybe the methods used were not appropriate for the given datasets, and perhaps using different approaches and

inputs could have yielded better results. Also, due to long computational times we did not manage to evaluate our hyperparameters to the fullest, only effects of a few hyperparameters were analyzed, again maybe more time spent on the hyperparameters evaluation could have produced better results.

The technologies developed in this research are not limited to only the problems discussed here, but can be applied in other fields and industries. Change-point models for example can be used for various purposes as mentioned in Chapter 3.1.3, i.e. validation of model assumptions, assessment and monitoring of safety critical processes, and the validation of an untested scientific hypothesis. Change-point models can also be used in other fields as well, including, bioinformatics applications, finance application, network traffic analysis, climatology applications, oceanography applications and detection of malware within software. On the other hand, sequence modeling can be used for problems such as activity recognition, speech recognition, chatbots, sentiment classification, DNA sequence analysis, music generation, image generation, text generation etc. Time-series forecasting methods have plenty of applications as well, with one the most popular ones being weather forecasting, and stock and market forecasting.

References

A. Aziz. Measuring value at risk (var) for oil and gas company. *Unknown Journal*, 2014.

F. Bahari and S. Elayidom. An efficient crm-data mining framework for the prediction of customer behaviour. *Procedia Computer Science 46: 725 731*, 2015.

D. Barry and J. Hartigan. Product partition models for change point problems. *The Annals of Statistics*, 1992.

D. Barry and J. Hartigan. A bayesian analysis for change point problems. *Journal of the American Statistical Association, Vol. 88, No. 421 (Mar., 1993), pp. 309-319*, 1993.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks, 5(2):157-166*, 1994.

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research, 13(Feb):281-305*, 2012.

J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl. Algorithms for hyper-parameter optimization. *In Advances in Neural Information Processing Systems, pages 2546-2554*, 2011.

M. C. Bishop. Pattern recognition and machine learning, volume 1. *Springer New York*, 2006.

J. V. Braun and H. Muller. Statistical methods for dna sequence segmentation. *Statistical Science, 13(2):142162*, 1998.

D. Britz. Recurrent neural networks tutorial, part 1 introduction to rnns. *Artificial Intelligence, Deep Learning, and NLP*, 2015.

R. G. Brown. Smoothing, forecasting and predicition of discrete time. *Smoothing, Forecasting and Predicition of Discrete Time*, 1963a.

R. G. Brown. Smoothing, forecasting and predicition of discrete time. 1963b. Available at `https://books.google.co.za/books?id=XXFNW_QaJYgC`.

J. Brownlee. Results from comparing classical and machine learning methods for time series forecasting. *Deep Learning for Time Series*, 2019. URL `https://machinelearningmastery.com/findings-comparing-classical-and-machine-learning-methods-for-time-series-forecasting/`.

F. Buttle and Francis. Customer relationship management : concepts and tools. *Unknown Journal*, 2006.

E. Carlstein, H. G. Muller, D. Siegmund, and editors. Change-point problems. *Institute of Mathematical Statistics Lecture Notes.*, 1994.

J. Chen and A. K. Gupta. Parametric statistical change point analysis. *Birkhauser*, 2000.

S. Chib. Estimation and comparison of multiple change-point models. *Journal of Econometrics 86 (1998) 221-241*, 1998.

K. Cho, van B Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation:encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

F. Chollet and J. Allaire. Deep learning with r. *Unknown Journal*, 2018.

C. Christopher. The analysis of time series: Theory and practice. 1975.

C. Chung, K. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Unknown Journal*, 2014.

A. E. Clark and I. Durbach. Using bayesian change-point models to assess changes in customer loyalty over time. *Department of Statistical Sciences, University of Cape Town*, 2014.

J. Cochrane. A brief parable of over-differencing. *Unknown Journal*, 2018.

D. R. Computing. Time series analysis. *Unknown Journal*, 2005.

D. Cornelisse. An intuitive guide to convolutional neural networks. *Unknown Journal*, 2018.

R. A. Davis, T. C. M. Lee, and G. A. Rodriguez-Yam. Structural break estimation for nonstationary time series models. *Journal of the American Statistical Association , 101(473):223239*, 2006.

M. J. de Smith. Arma and arima (box-jenkins) models. *Statistical Analysis Handbook 2018 edition*, 2018. URL `http://www.statsref.com/HTML/?arima.html`.

F. Diebold and R. Mariano. Comparing predictive accuracy. *Journal of Business and Economic Statistics*, pages 13, 253–263, 1995.

G. Dlamini. Machine learning methods for individual acoustic recognition in a species of field cricket. *University Of Cape Town, Department of Statistical Sciences*, 2018.

A. Douglass, D. Quinlan, and R. Shaw. Moving average. *Unknown Journal*, 2012. URL `https://kgsmithfx.files.wordpress.com/2012/05/ma.pdf`.

I. Eckley, P. Fearnhead, and R. Killick. Analysis of changepoint models. *Department of Mathematics and Statistics, Lancaster University, Lancaster LA1 4YF*, 2011.

A. S. C. Ehrenberg. *The Pattern of Consumer Purchases*. Wiley for the Royal Statistical Society, 1959. URL `https://doi.org/10.2307/2985810`.

J. Elman. Finidng structure in time. *Cognitive Science*, 1990.

J. B. Elsner, F. N. Xu, and T. H. Jagger. Detecting shifts in hurricane rates. *Journal of Climate*, page 17:26522666, 2004.

P. Fryzlewicz and S. Rao. Basta: consistent multiscale multiple change-point detection for piecewise-stationary arch processes. *Unknown Journal*, 2011.

R. Gangurde, B. Kumar, and S. Gore. Building prediction model using market basket analysis. *International Journal of Innovative Research in Computer and Communication Engineering*, 2005.

R. Garland and P. Gendall. Testing dick and basu's customer loyalty model. *Australasian Marketing Journal (AMJ)*, 2004.

A. Graves. Supervised sequence labelling with recurrent neural networks. *volume 385. Springer*, 2012.

H. Green and G. Milne. Assessing model perfomance: The gini statistic and its error. *Data Marketing and Customer Strategy Management*, 2010.

A. K. Gupta and J. Chen. Detecting changes of mean in multidimensionalnormal sequences with applications to literature and geology. *Computational Statistics*, page 11:211221, 1996.

D. Harvey, S. Leybourne, and P. Newbold. Testing the equality of prediction mean squared errors. pages 13(2), 281–291, 1997.

S. Haykin. Neural networks: A comprehensive foundation. *Pentrice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition*, 1998.

R. Henderson and J. N. S. Matthews. An investigation of changepoints in the annual number of cases of haemolytic uraemic syndrome. *Applied Statistics*, page 42:461471, 1993.

D. Hinkley. Inference about the change-point in a sequence of random. *Unknown Journal*, pages 1–17, 1970a.

D. V. Hinkley. Inference about the change-point in a sequence of randomvariables. *Biometrika*, 1970b.

S. Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma thesis*, 1991.

C. Holt. Forecasting trends and seasonal by exponentially weighted averages. 1957.

D. Hubel and T. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 1962.

M. Husken and P. Stagge. Recurrent neural networks for time series classiffication. *Neurocomputing 50 (C):223-235*, 2003.

R. J. Hyndman and G. Athanasopoulos. Forecasting: Principles and practice. 2013. URL `http://otexts.com/fpp/`.

U. Jagare. Data science strategy for dummies. *Unknown Journal*, 2019.

K. Jarrett, K. Kavukcuoglu, and Y. LeCun. What is the best mult-istage architecture for object recognition? *In Computer Vision, 2009 IEEE 12th International Conference on, pages 2146-2153. IEEE*, 2009.

R. Jaxk, J. Chen, X. L. Wang, R. Lund, and L. QiQi. A review and comparison of changepoint detection techniques for climate data. *Journal of Applied Meteorology and Climatology*, page 6:900915, 2007.

D. Kantu and I. Durbach. Early warning systems for detecting changes in marketing metrics over time. *SAMRA*, 2013.

U. Karn. An intuitive explanation of convolutional neural networks. *Unknown Journal*, 2018.

R. Killick, I. A. Eckley, P. Jonathan, and K. Ewans. Detection of changes in the characteristics of oceanographic time-series using statistical change point analysis. *Unknown Journal*, 2010.

S. Kostadinov. Understanding gru networks. *Unknown Journal*, 2017.

D. Kriesel. A brief introduction to neural networks. 2005a. URL `http://www.dkriesel.com/en/science/neural_networkshttp://www.dkriesel.com/_media/science/neuronalenetze-en-zeta2-2col-dkrieselcom.pdf`.

D. Kriesel. A brief introduction to neural networks. *Unknown Journal*, 2005b.

B. Kubiak and P. Weichbroth. Cross- and up-selling techniques in e-commerce activities. journal of internet banking and commerce. *Journal of Internet Banking and Commerce 15(3)*, 2010.

D. W. Kwon, K. Ko, M. Vannucci, A. L. N. Reddy, and S. Kim. Wavelet methods for the detection of anomalies and their application to network traffic analysis. *Quality and reliability engieneering international*, page 22:953969, 2006.

V. Lazarov and M. Capota. Churn prediction. *Unknown Journal*, 2007.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86(11):2278-2324*, 1998a.

E. Lee, C. Chen, S. Jeong, G. Chen, and W. Yuan. Forecasting the daily number of customers in each restaurant. 2016.

P. Lio and M. Vannucci. Wavelet change-point prediction of transmembrane proteins. *Bioinformatics*, page 16(4):376382, 2000.

Z. Lipton, J. Berkowitz, and C. Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

R. Loschi and F. Cruz. An analysis of the influence of some prior specications in the identification of change points via product partition model. *Computational Statistics and Data Analysis 39 (2002) 477 501*, 2019.

R. Loschi, F. Cruz, P. Iglesias, and R. Arellano-Valle. A gibbs sampling scheme to the product partition model: An application to change-point problems. *Computers and Operations Research*, 2001.

R. Luger. Exact tests of equal forecast accuracy with an application to the term structure of interest rates. *Monetary and Financial Analysis Department Bank of Canada Working Paper 2004-2*, 2005.

S. Makridakis, E. Spiliotis, and V. Assimakopoulos. Statistical and machine learning forecasting methods: Concerns and ways forward. 2018. URL `https://doi.org/10.1371/journal.pone.0194889`.

M. Marcellino, J. Stock, and M. Watson. A comparison of direct and iterated multistep ar methods for forecasting macroeconomic time series. 2005.

A. Mishra. Metrics to evaluate your machine learning algorithm. *Unknown Journal*, 2018.

D. C. Montgomery. Introduction to time series analysis and forecasting solutions set. *1st Edition*, 2014.

J. Neter, W. Wasserman, and G. Whitmore. Applied statistics. *Newton, Massachusetts*, 1988.

M. A. Nielsen. Neural networks and deep learning. 2015. URL `http://neuralnetworksanddeeplearning.com/index.html`.

A. Nosedal. Partial autocorrelation function, pacf. 2016.

E. S. Page. Continuous inspection schemes. *Biometrika*, pages 100–115, 1954a. URL `https://www.jstor.org/stable/2333009`.

D. S. Repo. Time-series analysis guide. *Data Science Knowledge Repo*, 2018.

A. Robinson and F. Fallside. The utility driven dynamic error propagation network. *University of Cambridge Department of Engineering*, 1987.

F. Rosenblatt. The perceptron, a perceiving and recognizing automaton. *Cornell Aeronautical Laboratory*, 1957.

F. Rosenblatt. Principles of neurodynamics. *Unknown Journal*, 1962.

T. Royston-Webb. Propensity modelling for business. *A Data Science Foundation White Paper*, 2018.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations. *Nature, vol. 323*, pages 6088, 533–536, 1986.

H. Sak, A. Senior, and A. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *Unknown Journal*, 2014.

A. Sambvani. Neural network glossary. *Unknown Journal*, 2018.

D. Scherer, A. Muller, and S. Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. *Artificial Neural Networks - ICANN 2010, pages 92-101*, 2010.

D. Schmittlein and R. Peterson. Customer base analysis: An industrial purchase process application. *Marketing Science, 1994, vol. 13, issue 1, 41-67*, 1994.

A. Scott and M. Knott. A cluster analysis method for grouping means in the analysis of variance. *Biometrics, 30:507512*, 1974.

B. Sharad, S. Siddharth, and J. Dipak. Customer lifetime value measurement. *Management Science*, pages 54. 100–112, 2008. URL `https://www.researchgate.net/publication/220534770_Customer_Lifetime_Value_Measurement`.

S. Shen, G. Li, and H. Song. Is the time-varying parameter model the preferred approach to tourism demand forecasting? statisical evidence. *Unknown Journal*, 2019.

A. N. Shiryaev. On optimum methods in quickest detection problems. *Theory of Probability and its Applications*, page 2651, 1963. URL `http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=tvp&paperid=4645&option_lang=eng`.

M. Smith and R. Agrawal. A comparison of time series model forecasting methods on patent groups. *CEUR Workshop Proceedings. 1353. 167-173.*, pages 1353. 167–173, 2015.

L. Spokoiny. Multiscale local change point detection with applications to value-at-risk. *The Annals of Statistics*, page 37:14051436., 2009.

P. Stagge and B. Sendho. An extended elman net for modeling time series. *Unknown Journal*, 1997.

S. Tavish. A must-read introduction to sequence modelling (with use cases). *Analytics Vidhya*, 2019.

U. Triacca. Comparing predictive accuracy of two forecasts: The diebold-mariano test. 2011. URL `http://www.phdeconomics.sssup.it/documents/Lesson19.pdf`.

P. Udom and N. Phumchusri. A comparison study between time series model and arima. 2019.

M. Valipour, M. Banihabib, and S. Behbahani. Parameters estimate of autoregressive moving. *Journal of Mathematics and Statistics 8*, pages 3: 330–338, 2019. ISSN 1549-3644.

M. van Greunen. Forecasting methods for cloud hosted resources, a comparison. 2015.

A. Viera and J. Garrett. Understanding interobserver agreement: The kappa statistic. *Research Series*, 2005.

S. Wang and W. Chaovalitwongse. Evaluating and comparing forecasting models. *Wiley Encyclopedia of Operations Research and Management Science*, 2011a. URL `https://www.researchgate.net/publication/313991989_Evaluating_and_Comparing_Forecasting_Models`.

S. Wang and W. Chaovalitwongse. Evaluating and comparing forecasting models. 2011b. URL `https://www.researchgate.net/publication/313991989_Evaluating_and_Comparing_Forecasting_Models`.

P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE, 78(10):1550-1560*, 1990.

P. R. Winters. Forecasting sales by exponentially weighted moving. *Management Science, vol. 6*, pages 3, 324–342, 1960.

M. Xu and J. Walton. Gaining customer knowledge through analytical crm. *Industrial Management and Data Systems*, 2005.

G. Yan, Z. Xiao, and S. Eidenbenz. Catching instant messaging worms with change-point detection techniques. *In Proceedings of the USENIX workshop on large-scale exploits and emergent threats*, 2019.

B. Zinyoni. Big data analytics framework for agriculture. *Unknown Journal*, 2017.

D. Zitzlsperger, T. Robbert, and S. Roth. Forecasting customer buying behaviour the impact of "one-time buyer. *University of Kaiserslautern*, 2015. URL https://www.researchgate.net/publication/267941016_Forecasting_Customer_Buying_Behaviour_The_impact_of_one-time_buyer.